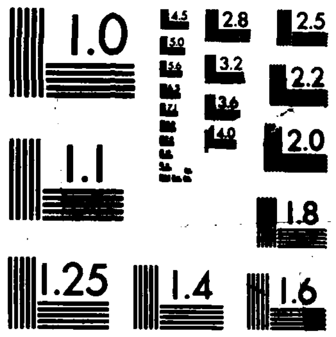
AD-A183 283 TOUCHSTONE: A CRITERIA DEVELOPMENT PROGRAM FOR GROUP DECISION SUPPORT SYSTEMS(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA R T WOOLDRIDGE ET AL. MAR 87 1/3

UNCLASSIFIED F/G 12/5 NL



MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS-1963-A

DTIC FILE COPY

2

NAVAL POSTGRADUATE SCHOOL

Monterey, California

AD-A183 203

DTIC ELECTE AUG 1 7 1987 S A D

THESIS

TOUCHSTONE:
A CRITERIA DEVELOPMENT PROGRAM FOR
GROUP DECISION SUPPORT SYSTEMS

by

Robert T. Wooldridge
and
Michael E. Neeley

March 1987

Thesis Advisor: Yuan Tung Bui

87 8 13 041

A183203

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School / 6b. OFFICE SYMBOL (If applicable) 54	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943 - 5000	7b. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943 - 5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION / 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11. TITLE (Include Security Classification)
TOUCHSTONE: CRITERIA DEVELOPMENT PROGRAM FOR GROUP DECISION SUPPORT SYSTEMS

12. PERSONAL AUTHOR(S)
Wooldridge, Robert T., and Neeley, Michael E.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Master's Thesis	FROM ______ TO ______	1987 March	234

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	GDSS, Delphi, Networking

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Group decision making utilizing the Delphi method can be a time-consuming and difficult procedure, especially when the required group membership is separated by great distances. This study designs and implements an automated group decision support system which may be employed by a single computer or a networking system.

This particular model is text-based as opposed to mathematical-based, a radical departure from the GDSS models currently in vogue. This program, TouchStone, successfully translates the Delphi method of criteria development to the computer. It is implemented in Turbo Pascal for the IBM-PC.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ XX UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL X. Tung Bui	22b. TELEPHONE (Include Area Code) (408) 646-2630	22c. OFFICE SYMBOL 54BD

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete. — SECURITY CLASSIFICATION OF THIS PAGE

TOUCHSTONE: A Criteria Development Program for Group Decision Support Systems

by

Robert T. Wooldridge
Commander, Nurse Corps, United States Navy
B.S.N, University of Virginia, 1969
M.A., Webster College, 1979
B.S., National University, 1985

and

Michael E. Neeley
Lieutenant, Medical Service Corps, United States Navy
B.S., University of Southern Illinois, 1979

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL
March 1987

Author: [illegible]
Robert T. Wooldridge

Author: [illegible]
Michael E. Neeley

Approved by: [illegible]
Xuan Tung Bui, Thesis Advisor

[illegible]
Nancy Roberts, Second Reader

[illegible]
Willis R. Greer, Jr., Chairman,
Department of Administrative Sciences

[illegible]
Kneale T. Marshall,
Dean of Information and Policy Sciences

ABSTRACT

Group decision making utilizing the Delphi method can be a time-consuming and difficult procedure, especially when the required group membership is separated by great distances. This study designs and implements an automated group decision support system (GDSS) which may be employed by a single computer or a networking system.

This particular model is text-based as opposed to mathematical-based, a radical departure from the GDSS models currently in vogue. This program, TouchStone, successfully translates the Delphi method of criteria development to the computer. It is implemented in Turbo Pascal for the IBM-PC.

Keywords: theses; computer program documentation; interfaces.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

The authors wish to acknowledge the support and assistance of the following people in the writing of this thesis.

Tung Bui
Wooldridge Family
LCDR David Moore, USN
Major Susan Quensel, USA
Captain Mark Hayes, USMC
LT Margaret A. Dunn, USN
LT Mary A. Woodbury, USN

I. INTRODUCTION

A. DEFINITION OF THE PROBLEM

In today's fast-paced world community, the logistics of assembling a group of experts for the purpose of resolving a particular problem has become a problem unto itself. Conflicting schedules, prohibitive distances, and the increasing frequency of group decision-making efforts are constant barriers to effective attacks on common problems. Even if such problems were easily surmountable, the importance and complexity of today's problems would require a decision based on the concensus of an expert group rather than the opinion of a single, stong-minded individual.

B. THE NEED FOR THE COMPUTERIZED GDSS

Managerial decision making has become increasingly more dependent upon computer-generated information. As a result, management is more cognizant of the capabilities and potentials of computer-based systems. The computer-based system has evolved from assisting individuals in making a decision to supporting and enhancing a wide range of group and organizational decisions. The question is how to effectively and efficiently design a distributed Decision Support System (DSS) to aid a group in defining, evaluating, modifying, and seeking consensus in deriving the criteria for a common problem. Recent literature in computer conferencing systems suggests that a computer-based Group

Decision Support System (GDSS) could:

1. Reduce tension due to face-to-face communications,
2. Promote equal participation, and
3. Favor free and creative generation of ideas.

C. SCOPE OF TOUCHSTONE

CO-OP, a program recently developed at the Naval Postgraduate School, Monterey, California, was designed to assist in the prioritization of previously-developed criteria. TouchStone, the program written as an adjunct to this thesis, is a prototype of a text-oriented, criteria-development system which may be utilized independently or as a "front-end" to the CO-OP program. Inasmuch as it is a prototype, there are necessary physical limitations to the number of problems, criteria, and people the system is designed to handle.

While both TouchStone and CO-OP are stand-alone systems, TouchStone offers a solid baseline of developed criteria upon which CO-OP builds, and from which it processes a decision, using mathematical modeling. TouchStone is a self-contained system, with an on-line, on-screen "users manual" that provides specific information based upon the user's position and status in the program. Use of TouchStone neither requires nor precludes the use of CO-OP, but these two systems complement each other in their methods of problem resolution.

D. ORGANIZATION OF THE THESIS

Inasmuch as this thesis is project-oriented, the actual text herein is minimal, limited primarily to a description of the background for, and the process of, putting the Delphi system on an electronic medium. The bulk of the information is contained in the source code found in Appendix E. It is the technique of implementing the text-orientation, the help screens, the communicative "Chatterbox", and the hierarchical text-manipulation, which is the essence of our efforts and our thesis. TouchStone is the thesis; this written effort is merely a support and a description of the true product of our research.

E. FOCUS OF THE THESIS

This thesis, and its accompanying computer program, focus on a particular aspect of group decision making. They develop a framework for guiding committee members to individually generate criteria for a collective problem, merge them together, and allow interactive negotiation and collective refinement of the set of criteria representing the problem. This concept is centered around the premise of the Delphi method of group decision-making and reflects the attempt of that method to provide anonymous and equal partnership in problem resolution. The peculiarity of the TouchStone system is its unique utilization of organized text processing without depending upon complex mathematical modeling to reach a conclusion.

F. OBJECTIVE

Our objective is to provide the proper mix of computer assistance and creative freedom for the TouchStone users as they attempt problem resolution with the Delphi method. The program is developed to support individuals and groups having expertise in the management field but not necessarily in the computer field. It is our intent to create an automated group decision-making tool that will take pressure, both real and imagined, away from the individual member serving on a committee, while not compromising the effectiveness of the committee as a whole. The system should allow the user to interact with other members of the committee, free from the effects of those members' actions, prejudices, and mannerisms.

II. THE DELPHI METHOD OF GROUP DECISION MAKING

A. BACKGROUND

Research literature on the subject of the Delphi methodology gives a wide variety of definitions and descriptions. The concept, developed primarily by the Rand corporation beginning in the late 1960's, has some fundamental building blocks common to most versions:

1. An individual who defines a particular problem.
2. A group of experts gathered together to resolve a particular problem.
3. A facilitator who collects the input from the experts, collates it, and gives the composite results back to the experts for further consideration.
4. Anonymity in the sense that the experts do not know the individual sources of the collective information, (although they may, in fact, know who else is in the group).

The purpose of the Delphi methodology is the elimination of external influences on group concensus and decisions.

> The idea is to improve the panel or committee approach in arriving at a forecast or estimate by subjecting the views of the individual participants to each other's criticism in way that avoids face-to-face confrontation. [Ref. 1]

It is by this technique that a free and open discussion of a problem may be implemented regardless of the personalities, ranks, or prestige of the participants. The solution to the problem, and little else, becomes the focus of the discussion.

B. COMPUTERIZATION OF THE DELPHI METHOD

Translating the Delphi method to the computer can be a relatively logical process. Building blocks 1 and 2 (see II, A.) are essentially unchanged; for building blocks 3 and 4, the computer replaces the human involvement. TouchStone refers to the individual defining the problem, as the 'problem invocator', and to the experts as the 'committee members'. Through the TouchStone program, the computer becomes the facilitator, collecting and collating the expert input. The anonymity of the experts is adequately maintained by the system to all but the problem invocator.

The major advantage to automating the Delphi method is time. The Delphi method is lengthy and cumbersome when executed on a committee of any significant size. The computer allows committee members to be located around the world and still to have instant access to the 'facilitator' at any time of day or night. In this manner, problems may be resolved in days instead of months, and the need to physically assemble a group of experts to resolve a problem is all but eliminated.

III. THE MODEL COMPONENT

A. MODEL BASE FOR GROUP DECISION MAKING: ALTERNATIVES VS. CRITERIA

> Our framework for DSS includes modeling and model usage as one of three basic components, completely integrated with data base and dialog capabilities. This full integration is necessary to support decision-making activities such as projection, deduction, creation, and comparison of alternatives. These activities require close interaction and rapid feedback between the decision maker and the computer, with strong and flexible control mechanisms. [Ref. 2, p. 276]

Alternatives are defined as the choices available for the resolution of a given problem; criteria are the guidelines to be used in making the final decision between those alternatives. TouchStone allows for the development of both of these important aspects of any decision, by allowing members to define, explain, discuss, re-define and agree upon a collective set of alternatives and criteria. Once this initial decision has been made, the remaining user responses and actions are the same for both. The initial decision of the committee member is to make the choice between developing alternatives or developing criteria.

The TouchStone system uses the Model-Dialog link as described by Sprague and Carlston in that six basic steps are utilized:

1. Invocation: user calls and starts the model
2. Parameter request: program requests data or parameters

3. Parameter collection: user supplies data or parameters

4. Interrupt: not usually available other than unrecoverable terminate (break) or pause.

5. Model completes run, notifies and presents results in a predefined format or report.

6. Return to step 1 for another cycle if desired. [Ref. 2, pp. 274-275]

B. PROBLEM INVOCATOR

The major design factor for TouchStone revolves around the creation of the problem and the responsibilities/limitations designated to resolve that problem. It was determined early in the research for this project that at least one person needed to be responsible for identifying the problems and for necessary housekeeping chores. We established this 'position' by looking at a normal face-to-face committee, and emulating the positions within the TouchStone System, making the "problem invocator" the committee chairman. The potential duties of the problem invocator have extensive ramifications and far reaching consequences. Initially, the invocator is responsible for naming the problem, providing a short but descriptive definition, and (optionally) expanding upon that definition to any length he feels necessary. He is also responsible for designating the committee members, adding and deleting members to any committee as indicated, and for removing completed problems from the system. Figures 16-19 exhibit screen menus with options available to the invocator. Final printouts of committee results and archival printouts of the Chatterbox file are under his purview (see Figure 29).

One of the most important decisions made by the invocator is that of committee member anonymity. The date/time/signature line in the Chatterbox (Chapter 6, paragraph D) may be modified to delete the automatic inclusion of the committee members' initials. In this manner, the interaction between members may be truly anonymous and in keeping with the spirit of the Delphi method of group decision-making. The use of the date/time signature stamp is two-fold, not only does it provide a reference point for committee members, it also allows the problem invocator to monitor the progress of the committee.

C. COMMITTEE MEMBER

The duties of the committee members are relatively simple to define. They are required to input their ideas and await further TouchStone system instruction at each level. Although the final product of their labors can be quite complex, the step-by-step methodology simplifies their efforts.

One of the major concerns of the Delphi method was that committee members be allowed to reach a concensus without being intimidated by the leader/invocator, or other committee members [Ref. 3]. Psychological research has shown that intimidation may occur by the tone of a person's voice, or even a casual glance from a superior [Ref. 4]. In the case of the TouchStone system, the invocator defines the problem, assigns members, and has total access over the system, but is unable to influence the committee members by any of his system actions. Also, the committee members are only able to influence other members by the strength of their ideas, not of their personality or position.

LEVEL 0:

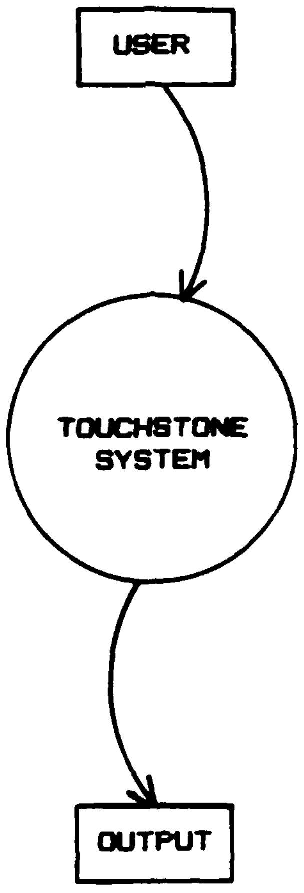


Figure 3.1 Data Flow Diagram

LEVEL 1:

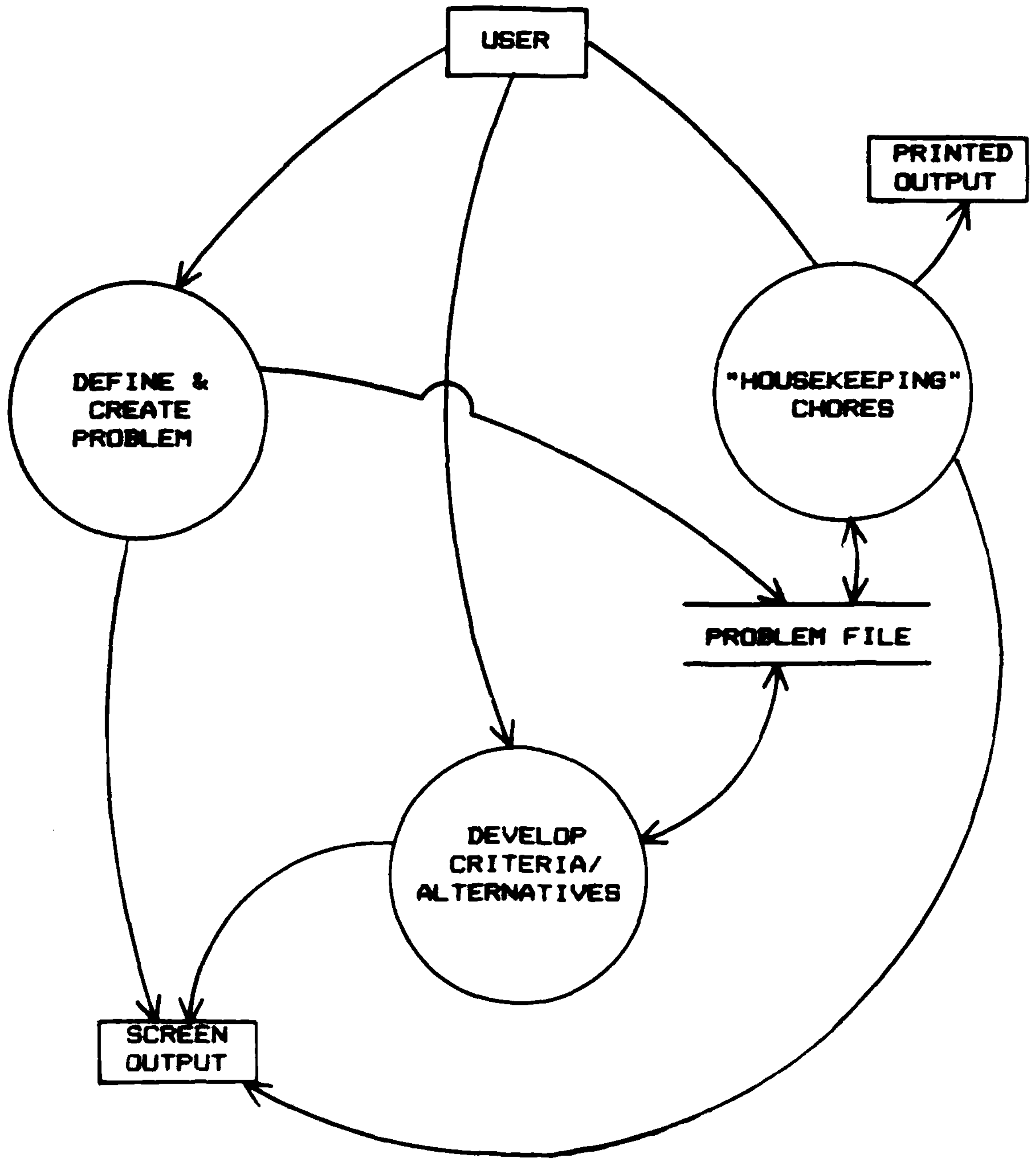


Figure 3.2 Data Flow Diagram

IV. <u>THE INTERFACE COMPONENT</u>

A. SCREEN DESIGN

The original concept for the screen design for Touch-Stone was to use a 3-window screen which would incorporate the problem definition, the Chatterbox, and the criteria manipulation. It soon became evident that this technique would not provide adequate space for any of the above-mentioned functions. The use of pop-up windows became the most reasonable alternative. Commercial software was researched, but it was felt that RAM resident windows did not provide adequate flexibility for context-sensitive help screens.

The use of multiple and/or "pop-up" windows was determined to be the most user-friendly method of providing communications and on-screen assistance. It was felt that simply refreshing the screen with the new screen, and then restoring it after the help or Chatterbox screen was through, was too distracting to the user. Employing windows allowed the user's main focus to remain on the problem screen, even when using the Chatterbox or the help screens.

The present screen design utilizes a number of separate, interactive screens. The main program uses a single box with the TouchStone logo at the top of the box. Each of the other screens is individually labeled, depending upon its function. Smaller boxes for the help screen, problem explanation and Chatterbox are layered onto the main screen. Any information overlaid by these boxes is restored when the box is removed. The boxes are carefully positioned for the express purpose of minimizing the amount of current information hidden by the overlay. (See Figures 36-38).

Screens are designed for maximum user effectiveness, keeping in mind, that a "busy" screen is often confusing. Menus are used as frequently as possible, limiting the number of choices to a minimum. The basic background colors are a light blue for all screens, with contrasting colors being utilized for special flags and pop-up windows. An example of this is the use of a red background for certain error messages.

One of the special features of TouchStone's screen design is the Odometer, which tracks and displays the user's relative position in the TouchStone decision making process. It also indicates a Chatterbox entry that the current user has not viewed. Located at the bottom of main the screens the Odometer also contains instructions for the use of the Function Keys. (See Figure 35).

B. DIALOGUE STYLE

As previously mentioned, the program is developed to support individuals and groups who are novices in computers. The use of "special function" keys is kept to a minimum, with clear definitions as to their usage displayed in the Odometer. Thus the simplicity of TouchStone eliminates the necessity for a manager or CEO to use a "computer chauffeur" for data input.

C. ON-LINE ASSISTANCE

Program assistance from TouchStone is provided in two forms, the "Introduction" screen and the "Help" screens. The "Introduction" screen is an option presented at the beginning of each TouchStone session, and contains a general, 4-page overview of the program.

The initial idea for the Help Screens was to implement an "automatic" screen, one which would appear when appropriate, without user action. Three categories of user expertise were defined, with corresponding levels of pop-up help windows. The user would indicate his ability level at the beginning of each session, following which the context-sensitive on-line help screens would appear as the <u>programmers</u> felt necessary. Subsequent research revealed that this idea was neither feasible nor desirable, from either the programmers' or the users' standpoint.

The present design of the "Help" screens for TouchStone follows the basic premise used by some of today's more popular software. A single function key (F-1) accesses one of the many pre-written help screens. Each screen is coded for access depending upon the user's location within the program. In this manner, the help screens remain current with the user and do not require a complex set of keystrokes on the user's part for access. The "help" text is frequently larger than the size of the screens, and a scrolling capability is implemented to compensate for this discrepancy.

V. THE DATA COMPONENT

A. DATA STRUCTURE/MANAGEMENT

The primary purpose of TouchStone, that of criteria/ alternative development, forces it to rely almost completely on the manipulation of text rather than data. The data component of TouchStone functions as a vehicle for flags and arrays. Each individual user of TouchStone is given a separate file for each problem to which he is assigned. That file contains the user name, the problem name, the current status of the user within that problem, and the criteria/alternatives that have been developed. When this file is created, an entry is made in the "master" file. Conversely, when a problem is concluded and the user files deleted, the master file is updated accordingly. These are the files dealing with text/data manipulation. Files utilized by the help screens, Chatterbox, and problem explanation screens are all text files. The help screen files have been created by the programmers; the problem explanation files is (optionally) created by the problem invocator at the time a new problem is defined; the Chatterbox files are created and updated each time the Chatterbox is used. The problem invocator has the option to print out the Chatterbox files at any time he so desires.

Data Management concerns itself with the recording, editing, and manipulation of text input for criteria and alternatives. Data management for TouchStone is based upon the complex alliance of two fundamental cornerstones: Flags and Arrays. The flags provide a "location map" for all members on a committee, allowing the program (and the problem invocator) to accurately monitor the progress and status of each problem resolution. Arrays provide the structure necessary to contain and control the free-flow text input vital to creative thought. The algorithm used for the marriage of these two building-blocks gives a large degree of freedom to the user while maintaining the structured environment required by the computer.

The manipulation of data is handled mainly by the extensive use of arrays. Data is initially input directly into a file. On the next user-access this data is brought up in the form of an array. This technique allows the sorting of individual files and, when required, the collating of multiple-user files. It also permits the user to 'edit' the text while reviewing his individual files. When multiple-user files are collated, duplicate records are eliminated, and the array replaces the original file with a new, composite file of criteria.

Manipulating text data from a variety of individuals calls for the use of an intricate series of flags. Each committee member's file has a flag-set based on the position of that file within the program. At certain points, continuation in the program is dependent upon the flag set of all other members in the committee. In addition, overseeing the progress of each problem resolution is an important task of the problem invocator. For these reasons, a separate master file was conceived, containing each problem name, each member of the committee dealing with each problem, and the current status of each member within a given problem.

The unique procedure "GetTheKeys" provides a variety of options for the system. Each keystroke is processed individually allowing the length of the input to be varied by the calling procedure. In that manner the user is prevented from entering data whose length is in excess of the size of the data field. The possibility of inputting a string of 60 characters, when the data field was only 10 characters long, is thereby eliminated. The reading of each keystroke also allows the function keys to be accessed at any time during the program, and during the review and editing of the text portion of the program, the special functions of the numeric keypad (i.e. arrows and paging keys) are activated.

An important feature of the data management of Touch-Stone is that it works in background mode, manipulating data, opening and loading files, and functioning as a system controller. It is an typical example of the "Black Box" in action. The user inputs data and receives results while the intricate process of weaving the input into a proper output goes largely unnoticed.

VI. THE COMMUNICATION COMPONENT

A. OVERVIEW

A main focus of TouchStone is communication--communication among users, communication between the user and the problem invocator, and user communication with the program itself. Without this intricate network of communication, the entire fabric of Touchstone would be lost.

B. TEXT EDITING

Inasmuch as TouchStone is highly involved in text manipulation, a variety of techniques in performing this manipulation was necessary to achieve our overall purpose. Once again, it was our goal to provide as much freedom as possible for the user while maintaining the necessary degree of system integrity. The concept of using a form of wordprocessing to input data is expected to be the most "user-friendly" method of inputting and manipulating data. Each keystroke is read and manipulated by our program. This practice allows the function keys and special keys to be programmer-defined and available throughout the system. Also, the on-line help-screens are automatically provided, progressing throughout the program.

Word-processing indicates the capability to block copy, move text, read to and from files, as well as text manipulation. TouchStone's version of "word-processing" is really a text editor, allowing for text input, erasure, scrolling, paging, and home/end-of-file movement. Three specific versions of text-editing are utilized in Touch-Stone, each necessitated by the very different conditions under which it is used.

The expanded problem explanation used by the problem invocator is a straight text editor employed when a problem is first described. Once invoked, a detailed explanation is written to file for later recall by the committee members. Full text manipulation is possible only by the invocator; committee members are limited to a read-only status. In this manner, only the problem invocator has the ability to define the problem, ensuring that each committee member is using the same baseline information.

Although previous Chatterbox entries are available for review, text editing in the Chatterbox is available only at the specified point at the end of the file. Action in the review mode is limited to scrolling and paging. Once an entry has been saved, it is not available for editing. By limiting editing access to the entry being made, a "permanent" record of Chatterbox entries may be made.

Text-editing in the main section of the program is limited to single-line input. The length of each line is location sensitive and specifically defined. This method allows for a wide range of functions, including the constant access to help and Chatterbox screens, as well as the ability to input string and numerical variables employing the same procedural call. Elimination of all "READ" and "READLN" calls was the unique contribution of this procedure and the basis for an increased elegance in programming.

C. HELP SCREENS

Help screens are important for the system to be informative. Help screens are discussed in Chapter IV, paragraph C.

D. PROBLEM EXPLANATION SCREEN

The problem invocator communicates with the committee members via the "problem explanation" box, accessed with the F-2 key. During the initial creation of the problem, the invocator is prompted to give a detailed explanation of the new problem. If he elects to do so, a file of up to 100 text lines is made available to him. The committee member then has a custom-made information file for each problem on which he is working. Text manipulation is "read" and "write" for the problem invocator (at the time of problem creation only) and "read-only" for the committee member. Since the problem explanation may be considerably larger than the problem explanation screen, scrolling and page up/down features are available to the user.

E. THE CHATTERBOX

The primary purpose of the Chatterbox is to promote the informal exchange of information among committee members. It has remained unchanged in its basic concept throughout the design and coding. However, of the many technical enhancements considered, those implemented were based primarily upon user acceptance.

Chatterbox differs from a conventional notepad in a number of ways. As mentioned before, in order to prevent "malicious" erasure of text, previous entries of text cannot be changed. Also, each individual problem has its own unique, automatically accessed, Chatterbox. Anytime the user leaves the Chatterbox, the file is saved unless no entry has been made. Any entry made in the Chatterbox is date/time/signature stamped providing an automatic record of the user. The problem invocator has the option of eliminating the signature from the viewed stamp for any given problem.

Location of the Chatterbox was the source of much discussion. The Chatterbox is located at the right-hand side of the screen, in order to leave important information residing in the main screen visible to the user. Ideally, it would be nice to provide a movable window; however, in this version, the location of the Chatterbox is fixed.

Designed to be used on a single computer or in a network, Chatterbox has a few unique features.

1) Only one person may write to Chatterbox at a given time, but more than more person may use it on a read-only basis.

2) The last 80 text lines of a given Chatterbox file are read into the Chatterbox array, with capability to add up to 40 lines of new text. However, a flag attached to the line counter prevents writing to any area except the last forty lines. In that manner, only new information may be edited.

3) One of the special features of the Chatterbox is to locate the user, upon re-entry, in the place

(time/date), where he last logged out of the Chatterbox. This feature allows him to check the messages that were entered after the last logout. Consequently, all new entries are immediately available for his review.

4) The line counter, in the upper right hand corner of the Chatterbox, allows for quick location reference when browsing.

5) Standardizing the line number between the read-write and read-only sections of Chatterbox made this delineation easier to implement. The appropriate placement of the text retrieval from the files was the primary key to controlling this procedure.

There were two specific issues which were considered, but rejected as part of the final design: 1) The imposition of time limits for a person using the Chatterbox was discussed but not implemented. It was felt that the use of a forty line limit on each entry was to be a sufficient constraint. 2) We also ruled out the possibility of importing data files into the Chatterbox. Such a situation would reduce the reading capability of the user, and fill the Chatterbox with excess information.

The Chatterbox is an integral part of the TouchStone system, being as important as the internal algorithms that aid the users in making a decision. Communication, as always, is vital to any decision-making process, and the Chatterbox enhances this aspect of the system.

VII. IMPLEMENTATION OF TOUCHSTONE

A. HARDWARE/SOFTWARE

TouchStone was developed on a Microsoft-based DOS computer with 640K RAM and a color card. TouchStone can be processed on a dual disk floppy drive system or a single floppy disk, with a hard disk system. Each floppy disk drive should be 360K RAM.

The Microsoft Disk Operating System utilized was version 3.1. The TouchStone System was written in Turbo Pascal version 3.01. No other software packages were employed in the final version of TouchStone. The system is comprised of four separate programs in the form of command files:

1) ATOUCH.COM
2) BTOUCH.COM
3) CTOUCH.COM
4) FLAGSET.COM.

These files are incorporated in a batch file called TS.bat. Each command file is basically a driver program, with numerous include files. These include files are listed in Appendix E. Documentation is done internally at the beginning of each procedure. Internal documentation lists the following:

Procedure name.
Program supported.
Local variables used.
Global variables used.
Arrays used.
Files accessed.

External Calls.
External filters (include files) used.
Where the procedure is called from.
Purpose of the procedure.

The effort expended (manhours) was as follows: system analysis and design, 100; research and thesis preparation, 150; coding, testing, and debugging: 700.

VIII. CONCLUSIONS

TouchStone, originally conceived as a criteria development tool for another DSS program ("CO-OP"), subsequently evolved into a stand alone program. As a non-mathematical, text-oriented GDSS, this program has entered a new area of computer support for making decisions. Although not thoroughly tested in a networking environment, the potential for such a use was an integral part of the design consideration and was incorporated in the final product.

TouchStone works. It provides a vehicle for criteria development in a group environment using the Delphi method, creating a novel technique of computer assistance. The objective of providing a proper mix of computer assistance and creative freedom in the explanation and analysis phase of the problem solving process, has been achieved.

APPENDIX A

DATA DICTIONARY

A, B, I, J, W, X, Y, Z: Various integer counters used throughout the system.

L, M and N: Integers that are summed and value passed to variable checkpoint.

ACTIVEPROBLEMFILE: file of PROBREC.

ALT: Single character used in identifying the file as an Alternative or Criteria, to be printed.

ALTERNATIVE: A single character, 'A' or 'C' for Alternatives or Criteria, used for assignment or comparisons.

ANONYMOUS: Boolean expression used in the chatterbox. When created, the problem invocator has the option to make communications anonymous from other committee members.

AUTHORIZED: Boolean expression, if true, allows the system to execute, if false, terminates the system.

CH, CHA: Single characters used for YES/NO type questions.

CHANGEFLAG: Boolean variable responsible for setting flags appropriately depending on whether the user is in "Alternatives" or "Criteria".

CHANGEREC: A single character used to confirm whether the problem is an Alternative or Criteria.

CHATRFILE: 12 character string denoting the chatterbox file to be used.

CHATOK: Boolean expression that controls the use of the chatterbox utility.

CHECKCHANGE: A single character used to confirm whether the problem is an Alternative or Criteria.

CHECKPOINT: Integer denoting the sum of the first three flags in this record. These records are sorted on this field to keep them in order according to the level of the data, i.e., 111 would equate a piece of data under the first major criteria, under the first sub-criteria.

CHECKSTATE: Is a single character used to track the user's position in the system.

CHKFLAG1, CHKFLAG2, and CHKFLAG3: Integers used to number the different levels of alternatives/criteria.

CHOICE: A single character, 'A' or 'C' for Alternatives or Criteria, used for assignment or comparisons.

CHT: Single character utilized for error trapping procedures.

CLEARIT: Integers used for tracking the arrays, advanced once for each record.

CODEARRAY: String of 12 characters used to encode an decode passwords.

CODENAME: String variable used for encoding and decoding user passwords.

COUNT, COUNTED, COUNTER: Integers used for tracking the arrays, advanced once for each record.

CRITARRAY: An array of the records in the format of CRIREC.

CRITDEF: String of 58 characters defining the above variable CRITNAME.

CRITERIA: Used in conjunction with the record CRIREC.

CRITLIMIT: Integer denoting the maximum number of alternatives/criteria allowed.

CRITNAME: String of 10 characters denoting criteria/alternatives name.

DATE: A string of 12 characters passed to a file as the day, month and year for tracking the last time a file was accessed.

DATELINE: String of 12 characters which gives the last date that the file was accessed.

DEFINITION: String of 58 characters which gives the short version of the problem definition.

DOUBLECOUNTED: An integer counter used during the merging of files process.

FILECHECK: Boolean expression used when checking the validity of a filename.

FILEDRIVE: Single character denoting the drive the data files reside on.

FLAGCHOICE: A string of 1 character used to set users problemflag.

FLAGCOUNT: Integers used for tracking the arrays, advanced once for each record.

FLAGEND: Integer that counts all files with the same problem name and the same flag setting.

FLAGGED: Single character used to check committee member status prior to merging files.

FLAG1: Integer denoting level 1, major criteria.

FLAG2: Integer denoting level 2, sub-criteria.

FLAG3: Integer denoting level 3, tertiary criteria.

HELPDRIVE: Single character denoting the drive the help files reside on.

HELPER: Single character that indicates the active help screen.

HELPSIZE: Integer parameter passed to determine the size of the helpscreen.

INPUTSTRING: Used with the variable STRINGARRAY, as a passed parameter to the procedure GetTheKeys.

INVOCATOR: A single character either a 'W' or 'M' used to determine whether the user is a problem invocator (M), or a committee member (C).

KEEPTOGETHER: An integer counter used during the sorting routine to keep the records in the various levels in the order in which they were entered.

KRITERIAFILE: file of CRIREC.

LIMID: An integer parameter passed to a procedure denoting the number of records in an array.

LIMMIT: Integer set to the maximum number of records in an array.

LINEMARK: Boolean expression used to advance line counter when displaying data on the screen.

MARKER: Integer used in conjunction with the gotoXY call when positioning data on the screen.

MEMBER: String of three characters which indicates that there is a file in the DOS directory with the extension using this members name.

MEMBERS: Used in conjunction with the record PROBREC.

MOVEOVER: Integer used in conjunction with the gotoXY call for positioning data on the screen.

MOVEX: Integer used with the gotoXY statement positioning data on the screen.

NAMES: Variable used with the record CRIREC and array CRITARRAY.

NAMESTRING: A string of three characters that is used as the extension when recalling the user's file.

NEWCRITLIMIT: Integer denoting the maximum number of alternatives/criteria allowed.

NEWLIMIT: An integer limiting the number of entries that can be made for alternatives/criteria.

NEWNAME: 3 character string used when verifying filenames.

NEWPROB: Single characters used for YES/NO type questions.

NEWSTRING: 12 character string denoting the file to be used.

NUM: Integers used for numbering the criteria/alternatives when displayed on the screen.

NUMMEMS: Integer that tracks the number of members on a particular committee. Minimum value of 2 and maximum value of 15.

ONCECOUNTED: A boolean expression used in the merging process.

PRINTONE: Boolean expression used when printing alternatives/criteria.

PROBARRAY: An array of the records in the format of PROBREC.

PROBLEM: String of seven characters which indicates that there is a file in the DOS directory beginning with this string.

PROBLEMFLAG: Single character used to track the status of the user who is logged on to TouchStone.

PROBNAME: A string of seven characters that is used as the first seven letters when recalling a user file.

PROBS: Variable used with the record PROBREC and array PROBARRAY.

PT1, PT2, PT3 and PT4: Integers used as points when defining the various windows used in the system.

QUITFLAG: Integer used in moving from level to level in the alternatives/criteria data entry.

QUITFLG1, QUITFLG2, QUITFLG3: Integers tracking the number of alternatives/criteria at the various levels.

RECOUNT: Integer used in positioning the pointer when writing to a users problem file.

SCROLLIT: Boolean expression that controls the use of the arrow keys, so that they may only be used during certain portions of the program.

SECNUM: Integers used for numbering the criteria/alternatives when displayed on the screen.

SELECTED: Integers used for tracking the arrays, advanced once for each record.

SHOWME: Integer used in moving from level to level in the alternatives/criteria data entry.

STARTMERGE: A boolean expression, that, when true allows all files with the same problem name to be merged into one.

STARTUP: Boolean expression used in several procedures to check the validity of the file requested or to check for duplication.

STATFLAG: Character that tracks where the user is in the system.

STRINGARRAY: An array of 1 to 59 characters, used in conjunction with the procedure GetTheKeys.

STOPGAP: Boolean expression used to stop alternatives/criteria input beyond a predetermined limit.

STOPPROG: Boolean expression, if true terminates a procedure or the entire program, depending on when it is toggled.

TEMPFILE: A temporary file using text vice records.

TEMPNAME: String variable used for encoding and decoding user passwords.

THRNUM: Integers used for numbering the criteria/alternatives when displayed on the screen.

TRACK1: Integer denoting number of records in an array.

USERCODE: 8 character code used to verify password.

WEEDDEF: Boolean expression used to activate the F3 key when the program goes past the problem selection stage.

APPENDIX B

FILE STRUCTURE

PROBREC: Is the master record that holds the following information on all of the problems in the system. The following variables comprise this record:

CHECKCHANGE: A single character used to confirm whether the problem is an Alternative or Criteria.

CHECKSTATE: Is a single character used to track the user's position in the system.

CHOICE: A single character, 'A' or 'C' for Alternatives or Criteria, used for assignment or comparisons.

DATELINE: String of 12 characters which gives the last date that the file was accessed.

DEFINITION: String of 58 characters which gives the short version of the problem definition.

MEMBER: String of three characters which indicates that there is a file in the DOS directory with the extension using this members name.

NUMMEMS: Integer that tracks the number of members on a particular committee. Minimum value of 2 and maximum value of 15.

PROBLEM: String of seven characters which indicates that there is a file in the DOS directory beginning with this string.

CRIREC: Is a record that is contained in a file in DOS. There is one file for each committee member for each specific problem. The record contains the following information:

CHECKPOINT: Integer denoting the sum of the first three flags in this record. These records are sorted on this field to keep them in order according to the level of the data, i.e., 111 would equate a piece of data under the first major criteria, under the first sub-criteria.

CRITDEF: String of 58 characters defining the above variable CRITNAME.

CRITNAME: String of 10 characters denoting criteria/alternatives name.

FLAG1: Integer denoting level 1, major criteria.

FLAG2: Integer denoting level 2, sub-criteria.

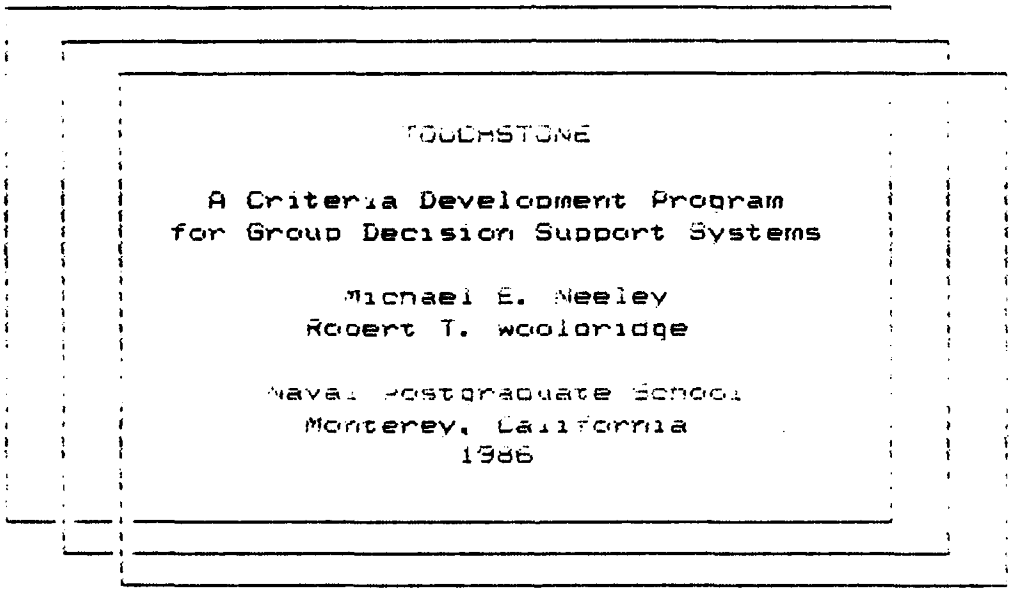
FLAG3: Integer denoting level 3, tertiary criteria.

STATFLAG: Character that tracks where the user is in the system.

APPENDIX C

SCREEN FORMATS

FIGURE 1
TITLE SCREEN



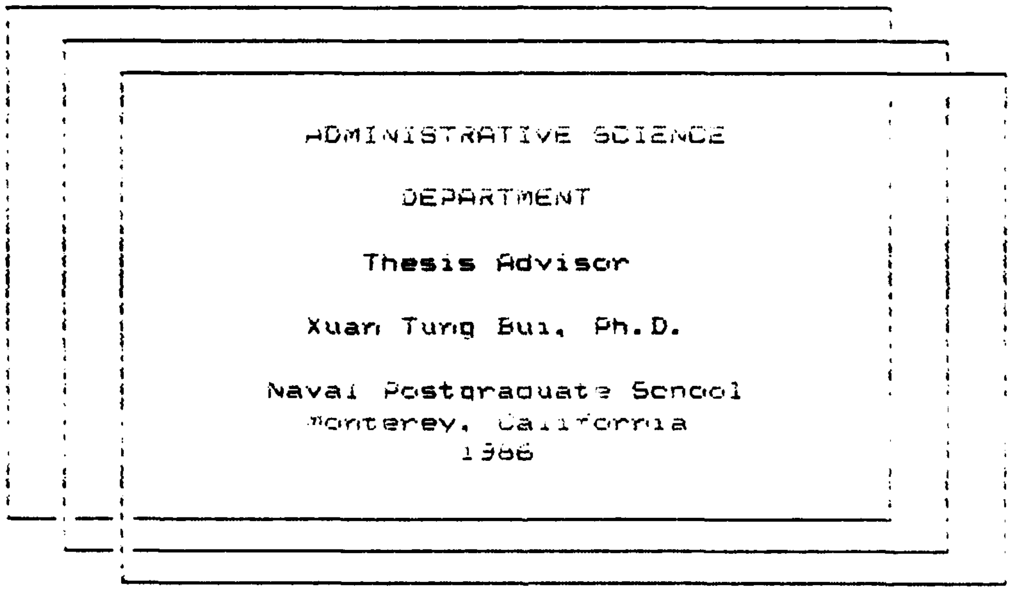
TOUCHSTONE

A Criteria Development Program
for Group Decision Support Systems

Michael E. Neeley
Robert T. Wooldridge

Naval Postgraduate School
Monterey, California
1986

FIGURE 2
THESIS ADVISOR SCREEN


ADMINISTRATIVE SCIENCE

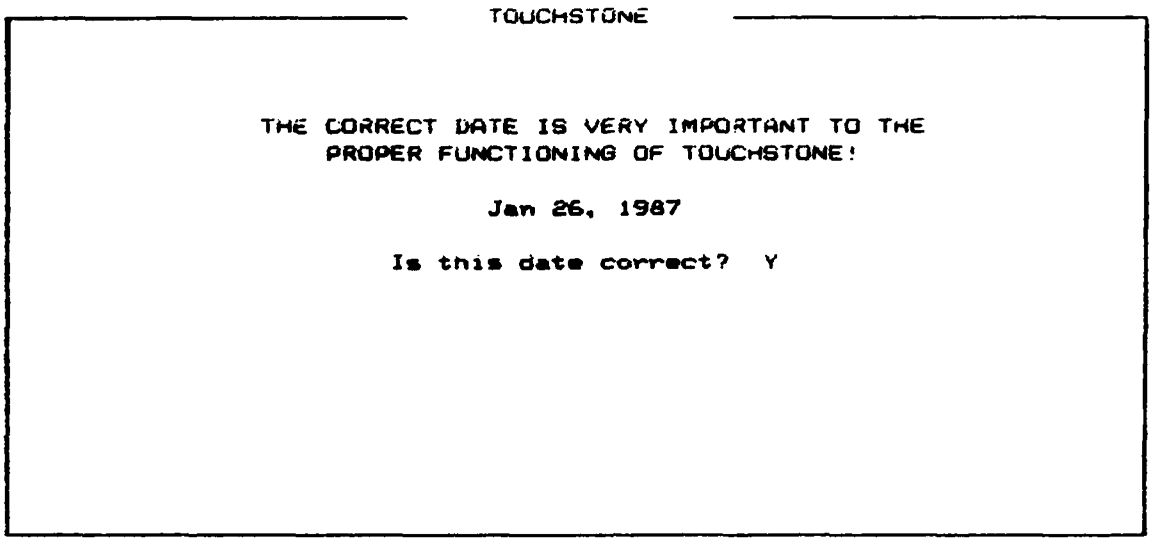
DEPARTMENT

Thesis Advisor

Xuan Tung Bui, Ph.D.

Naval Postgraduate School
Monterey, California
1988

FIGURE 3
DATE SCREEN

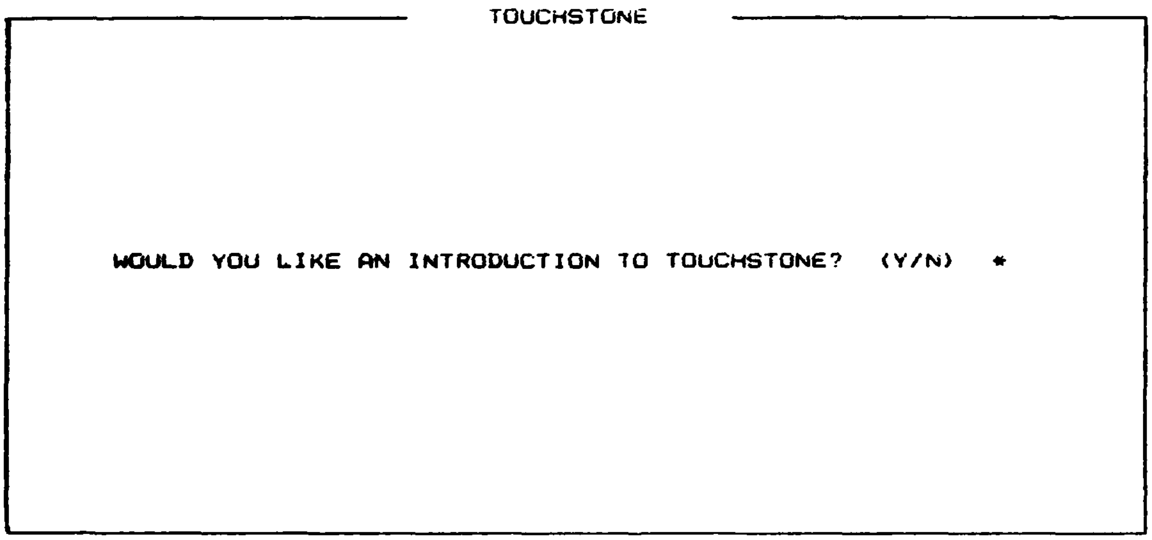

TOUCHSTONE

THE CORRECT DATE IS VERY IMPORTANT TO THE
PROPER FUNCTIONING OF TOUCHSTONE!

Jan 26, 1987

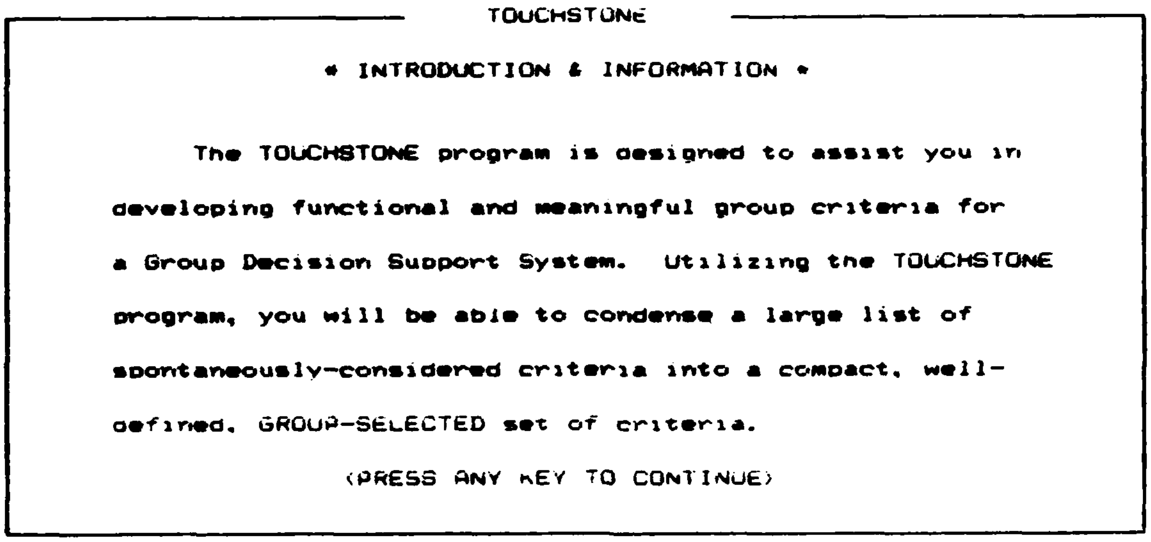
Is this date correct? Y

FIGURE 4
INTRODUCTION OPTION SCREEN


TOUCHSTONE

WOULD YOU LIKE AN INTRODUCTION TO TOUCHSTONE? (Y/N) *

FIGURE 5
INSTRUCTION SCREEN #1



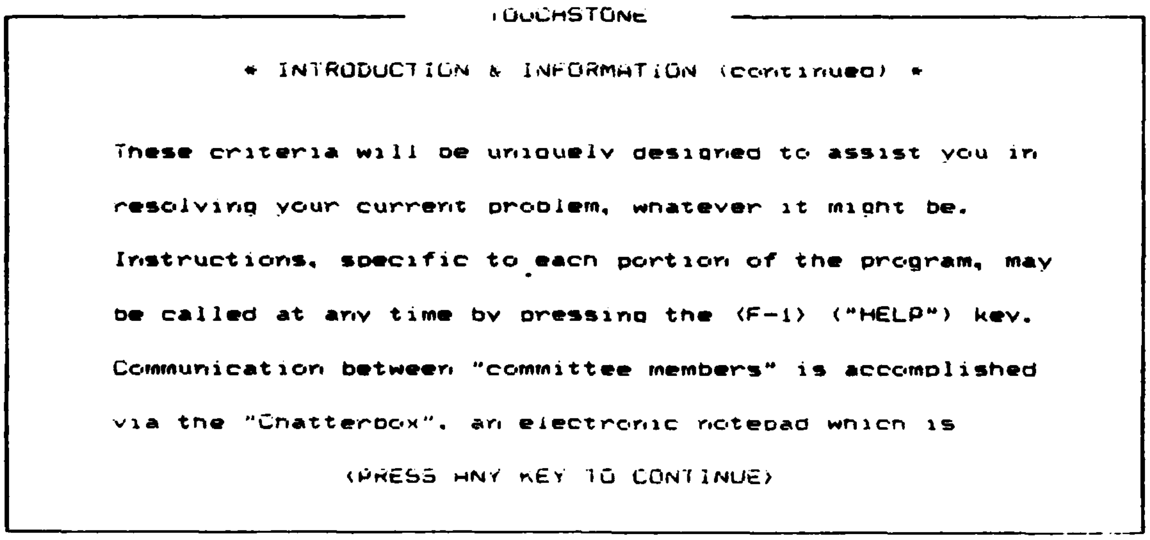
TOUCHSTONE

* INTRODUCTION & INFORMATION *

The TOUCHSTONE program is designed to assist you in developing functional and meaningful group criteria for a Group Decision Support System. Utilizing the TOUCHSTONE program, you will be able to condense a large list of spontaneously-considered criteria into a compact, well-defined, GROUP-SELECTED set of criteria.

(PRESS ANY KEY TO CONTINUE)

FIGURE 6
INSTRUCTION SCREEN #2



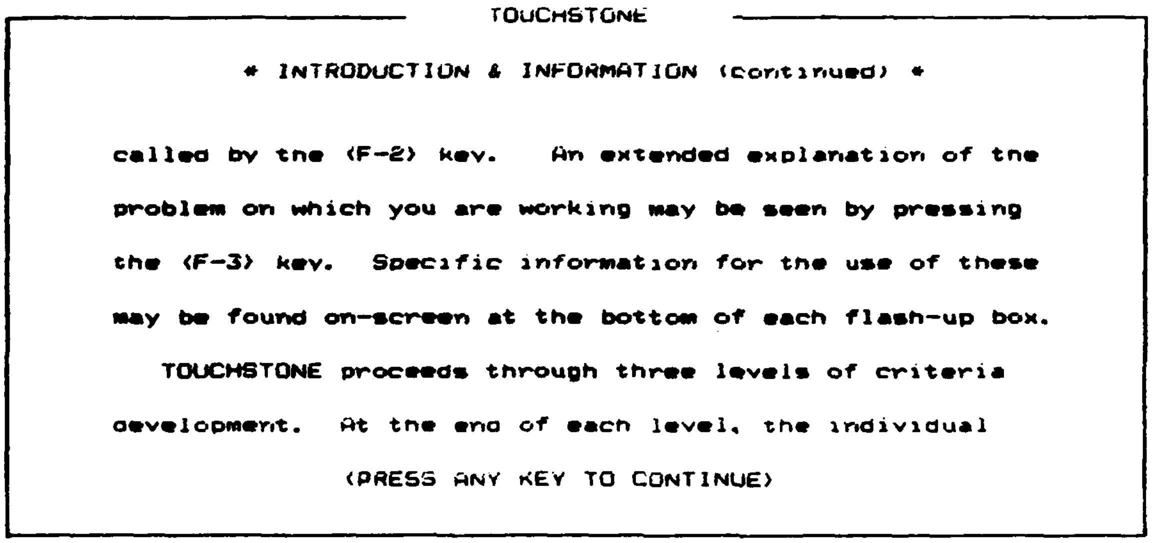
TOUCHSTONE

* INTRODUCTION & INFORMATION (continued) *

These criteria will be uniquely designed to assist you in resolving your current problem, whatever it might be. Instructions, specific to each portion of the program, may be called at any time by pressing the (F-1) ("HELP") key. Communication between "committee members" is accomplished via the "Chatterbox", an electronic notepad which is

(PRESS ANY KEY TO CONTINUE)

FIGURE 7
INSTUCTION SCREEN #3



TOUCHSTONE

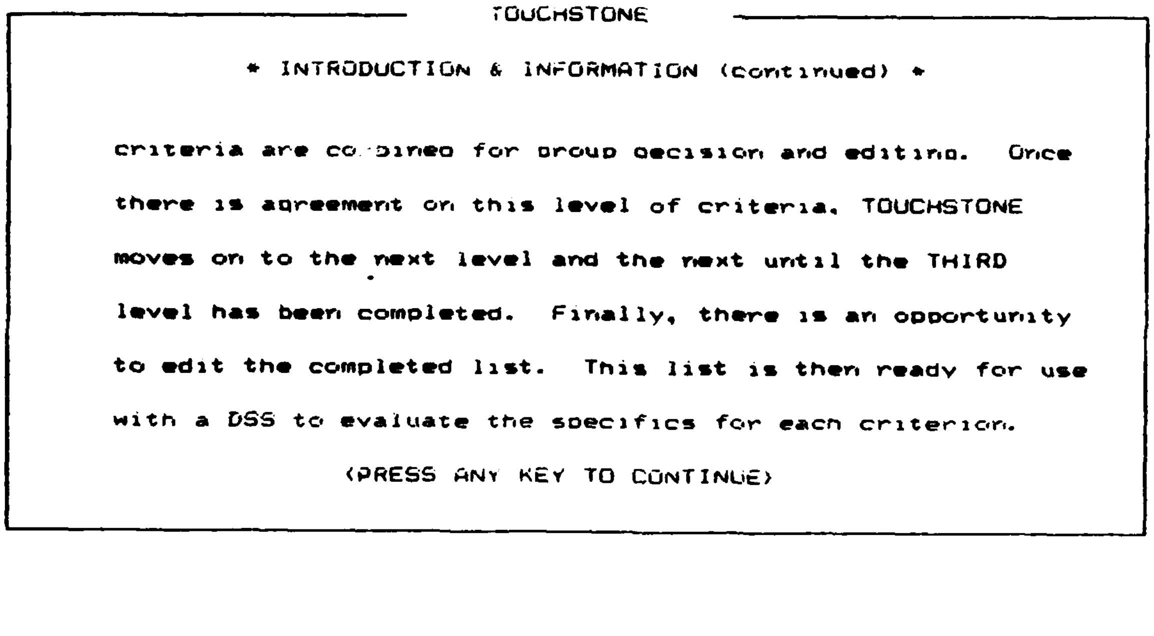
* INTRODUCTION & INFORMATION (continued) *

called by the (F-2) key. An extended explanation of the problem on which you are working may be seen by pressing the (F-3) key. Specific information for the use of these may be found on-screen at the bottom of each flash-up box.

TOUCHSTONE proceeds through three levels of criteria development. At the end of each level, the individual

(PRESS ANY KEY TO CONTINUE)

FIGURE 8
INSTRUCTION SCREEN #4



TOUCHSTONE

* INTRODUCTION & INFORMATION (continued) *

criteria are combined for group decision and editing. Once there is agreement on this level of criteria, TOUCHSTONE moves on to the next level and the next until the THIRD level has been completed. Finally, there is an opportunity to edit the completed list. This list is then ready for use with a DSS to evaluate the specifics for each criterion.

(PRESS ANY KEY TO CONTINUE)

FIGURE 9
FILE INITIALIZATION SCREEN

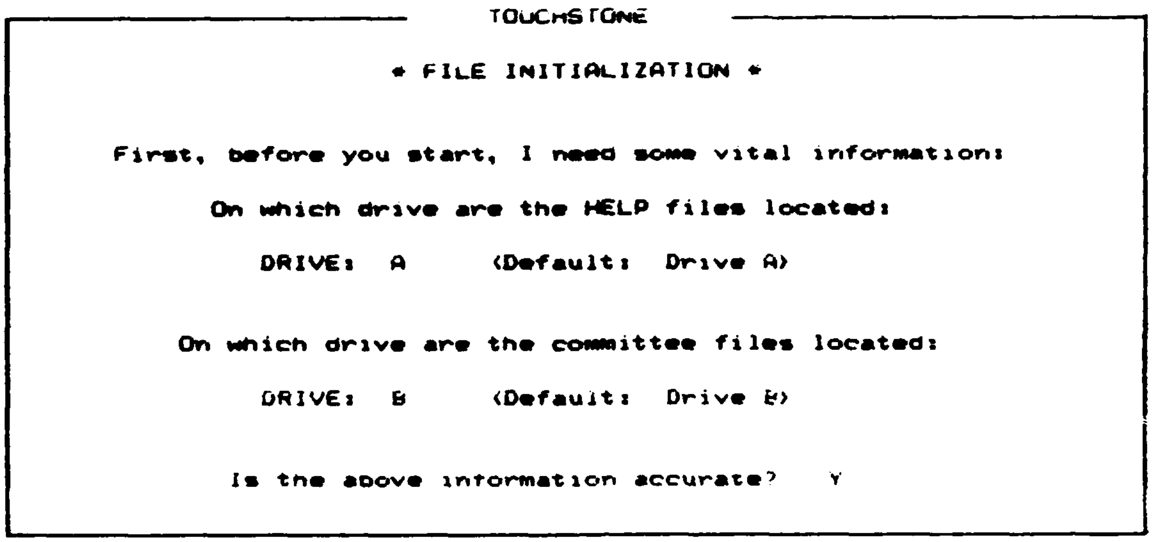

```
                                TOUCHSTONE

                          * FILE INITIALIZATION *

      First, before you start, I need some vital information:

              On which drive are the HELP files located:

                 DRIVE:  A      (Default:  Drive A)

          On which drive are the committee files located:

                 DRIVE:  B      (Default:  Drive B)

               Is the above information accurate?   Y
```

FIGURE 10
INITIALIZATION SCREEN FOR FIRST PROBLEM INVOCATOR

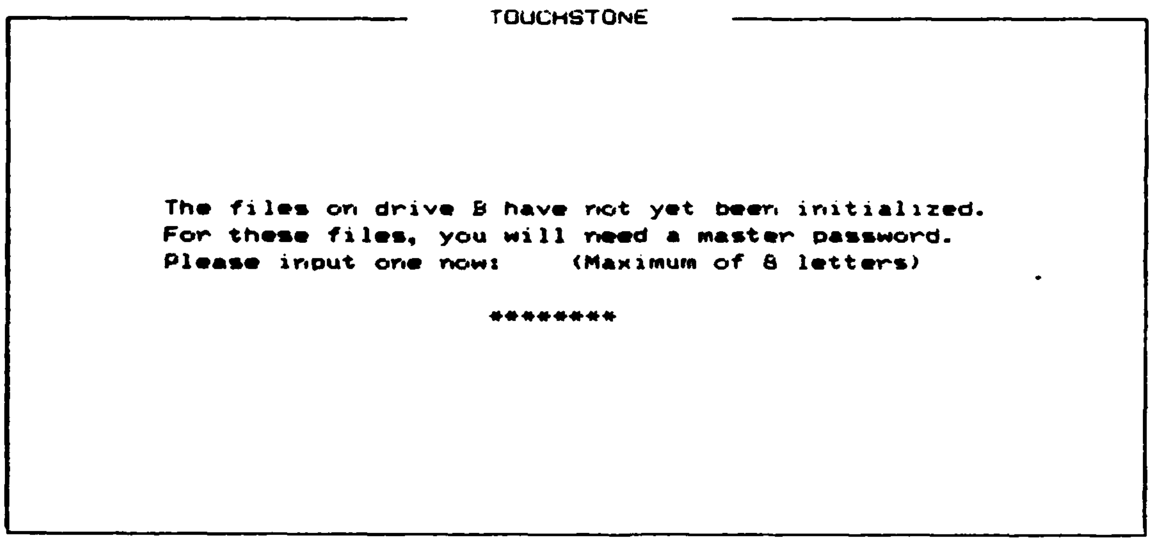
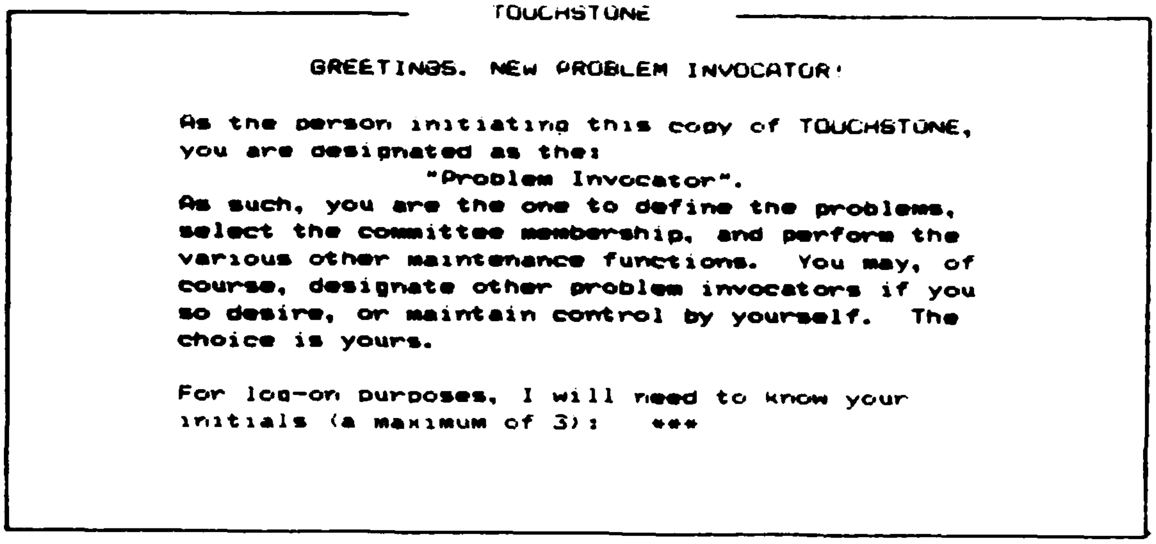
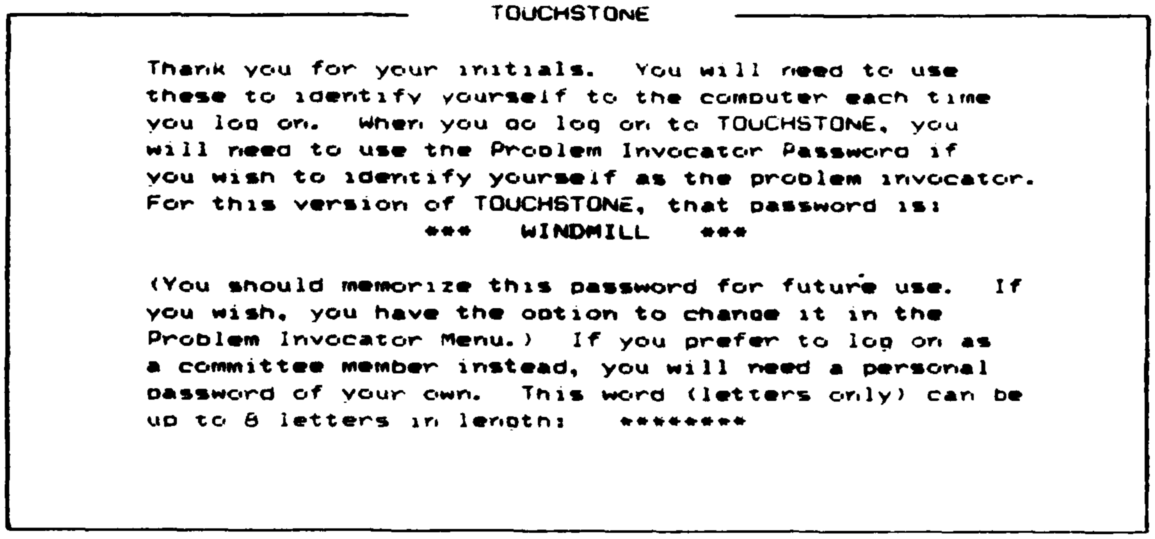

```
                                TOUCHSTONE

          The files on drive B have not yet been initialized.
          For these files, you will need a master password.
          Please input one now:    (Maximum of 8 letters)

                               ********
```

FIGURE 11
INTRODUCTION SCREEN (1) FOR FIRST PROBLEM INVOCATOR



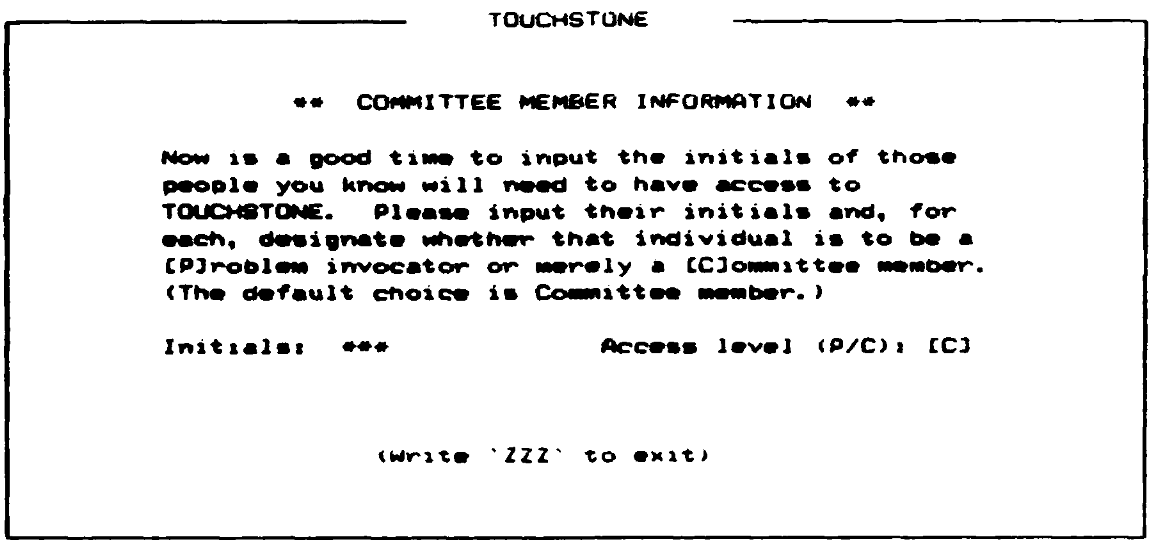
```
                    TOUCHSTONE

          GREETINGS. NEW PROBLEM INVOCATOR!

     As the person initiating this copy of TOUCHSTONE,
     you are designated as the:
                   "Problem Invocator".
     As such, you are the one to define the problems,
     select the committee membership, and perform the
     various other maintenance functions.  You may, of
     course, designate other problem invocators if you
     so desire, or maintain control by yourself.  The
     choice is yours.

     For log-on purposes, I will need to know your
     initials (a maximum of 3):    ***
```

FIGURE 12
INTRODUCTION SCREEN (2) FOR FIRST PROBLEM INVOCATOR



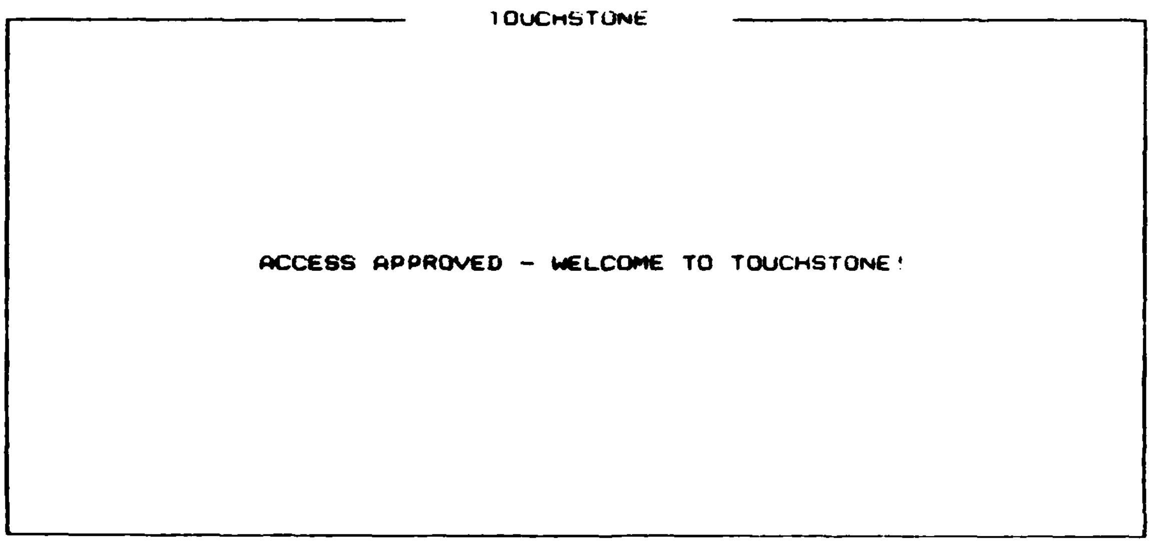
```
                    TOUCHSTONE

  Thank you for your initials.  You will need to use
  these to identify yourself to the computer each time
  you log on.  When you do log on to TOUCHSTONE, you
  will need to use the Problem Invocator Password if
  you wish to identify yourself as the problem invocator.
  For this version of TOUCHSTONE, that password is:
                  ***   WINDMILL   ***

  (You should memorize this password for future use.  If
  you wish, you have the option to change it in the
  Problem Invocator Menu.)  If you prefer to log on as
  a committee member instead, you will need a personal
  password of your own.  This word (letters only) can be
  up to 8 letters in length:    ********
```

FIGURE 13
INPUT COMMITTEE MEMBER/PROBLEM INVOCATOR INFORMATION



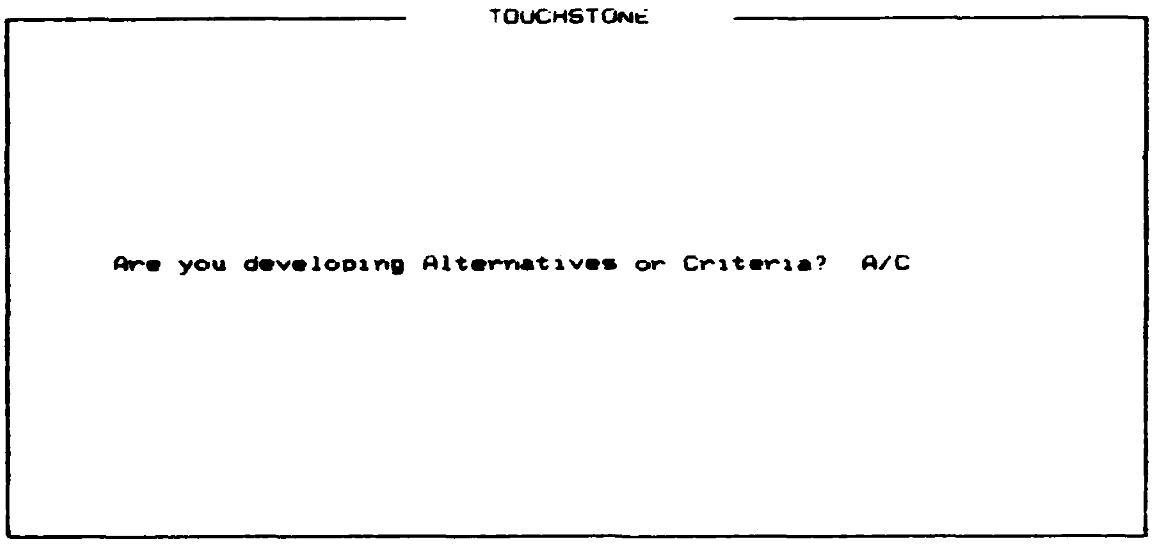
```
                ** COMMITTEE MEMBER INFORMATION **

Now is a good time to input the initials of those
people you know will need to have access to
TOUCHSTONE.  Please input their initials and, for
each, designate whether that individual is to be a
[P]roblem invocator or merely a [C]ommittee member.
(The default choice is Committee member.)

Initials:  ***              Access level (P/C): [C]


                   (Write `ZZZ` to exit)
```

FIGURE 14
ACCESS APPROVAL SCREEN



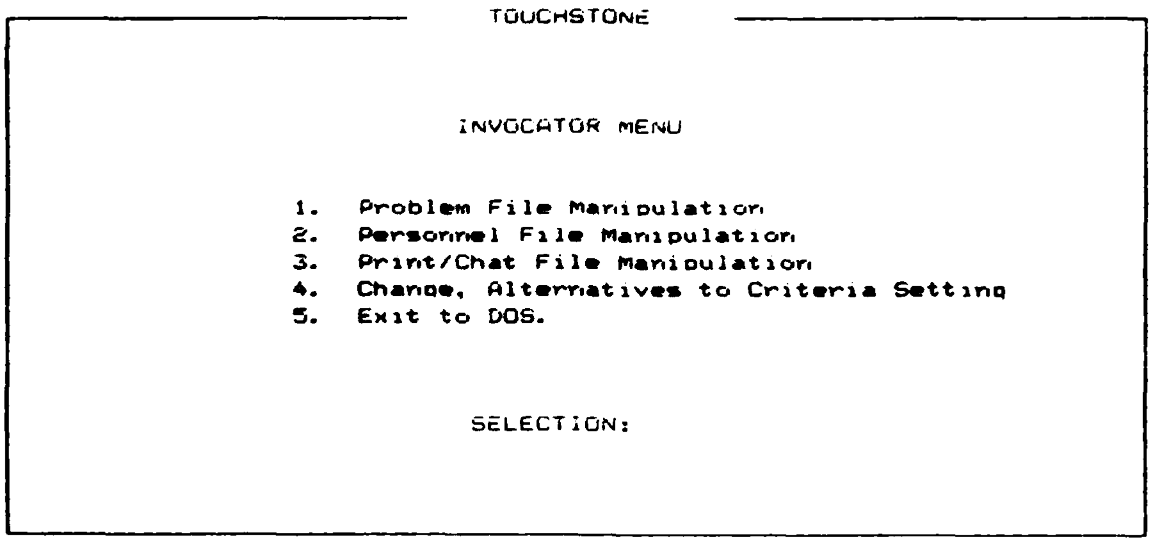
```
ACCESS APPROVED - WELCOME TO TOUCHSTONE!
```

FIGURE 15
ALTERNATIVE/CRITERIA CHOICE SCREEN



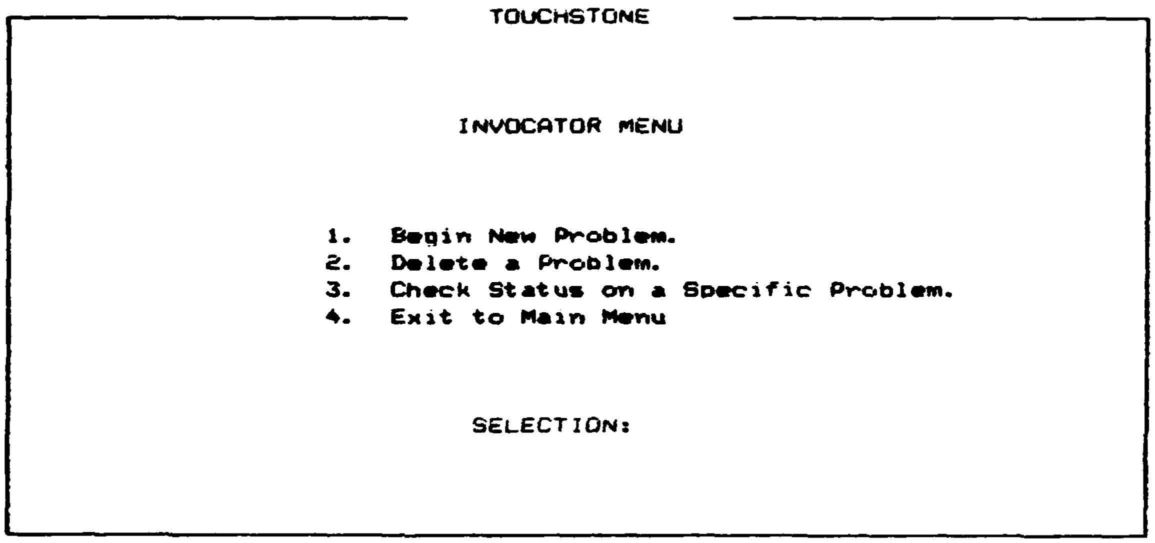
```
TOUCHSTONE

Are you developing Alternatives or Criteria?  A/C
```

FIGURE 16
MAIN INVOCATOR MENU SCREEN



```
TOUCHSTONE

INVOCATOR MENU

1.   Problem File Manipulation
2.   Personnel File Manipulation
3.   Print/Chat File Manipulation
4.   Change, Alternatives to Criteria Setting
5.   Exit to DOS.

SELECTION:
```

FIGURE 17
PROBLEM FILE MANIPULATION SCREEN



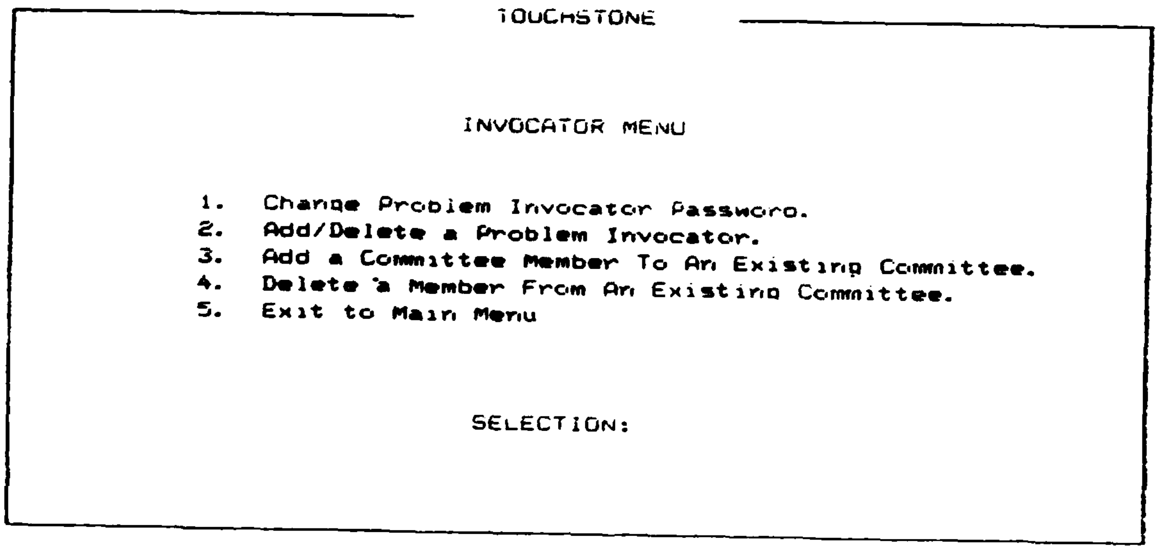
TOUCHSTONE

INVOCATOR MENU

1. Begin New Problem.
2. Delete a Problem.
3. Check Status on a Specific Problem.
4. Exit to Main Menu

SELECTION:

FIGURE 18
PERSONNEL FILE MANIPULATION SCREEN



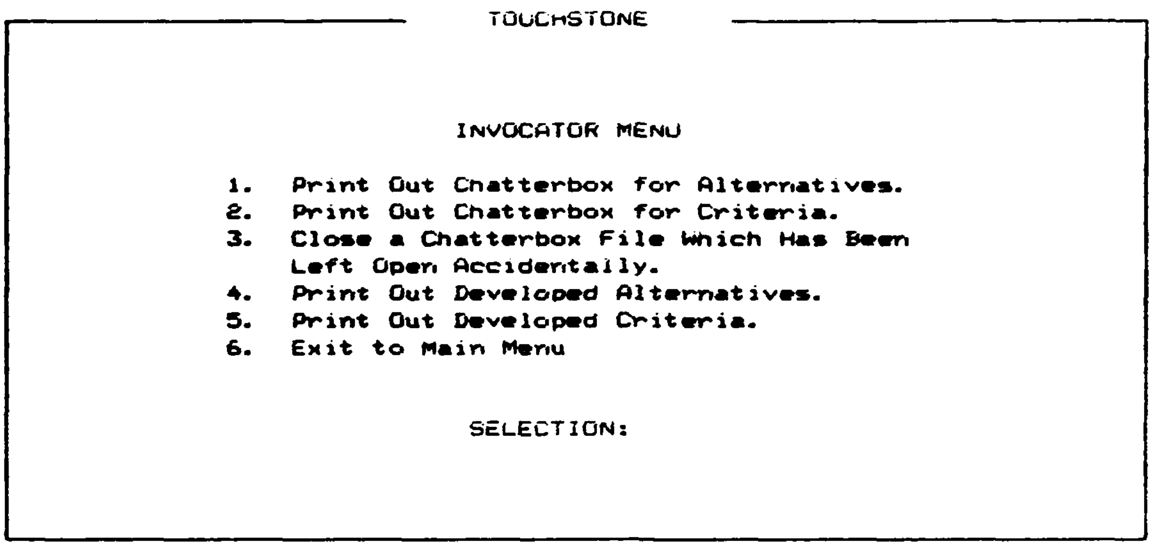
TOUCHSTONE

INVOCATOR MENU

1. Change Problem Invocator Password.
2. Add/Delete a Problem Invocator.
3. Add a Committee Member To An Existing Committee.
4. Delete a Member From An Existing Committee.
5. Exit to Main Menu

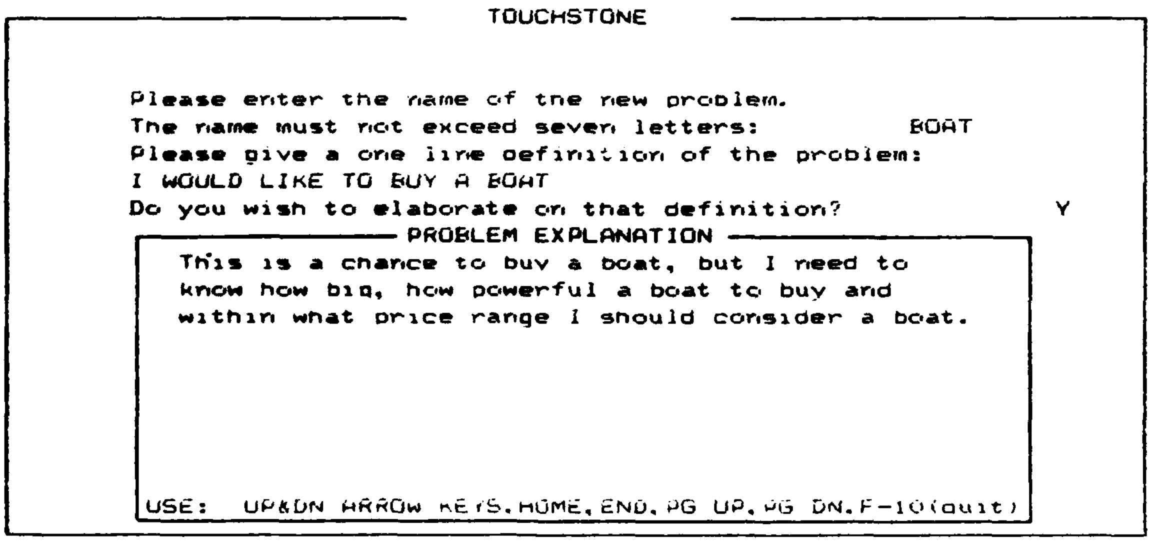
SELECTION:

FIGURE 19
PRINT/CHAT FILE MANIPULATION SCREEN



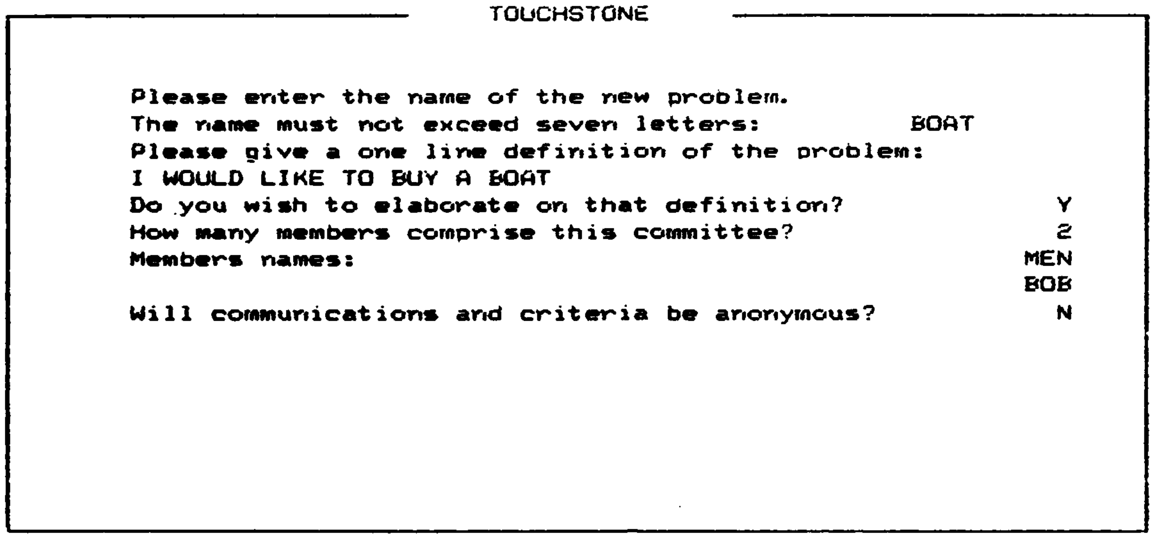
```
                         TOUCHSTONE

                       INVOCATOR MENU

        1.   Print Out Chatterbox for Alternatives.
        2.   Print Out Chatterbox for Criteria.
        3.   Close a Chatterbox File Which Has Been
             Left Open Accidentally.
        4.   Print Out Developed Alternatives.
        5.   Print Out Developed Criteria.
        6.   Exit to Main Menu

                        SELECTION:
```

FIGURE 20
PROBLEM CREATION SCREEN
(with PROBLEM EXPLANATION INSERT)



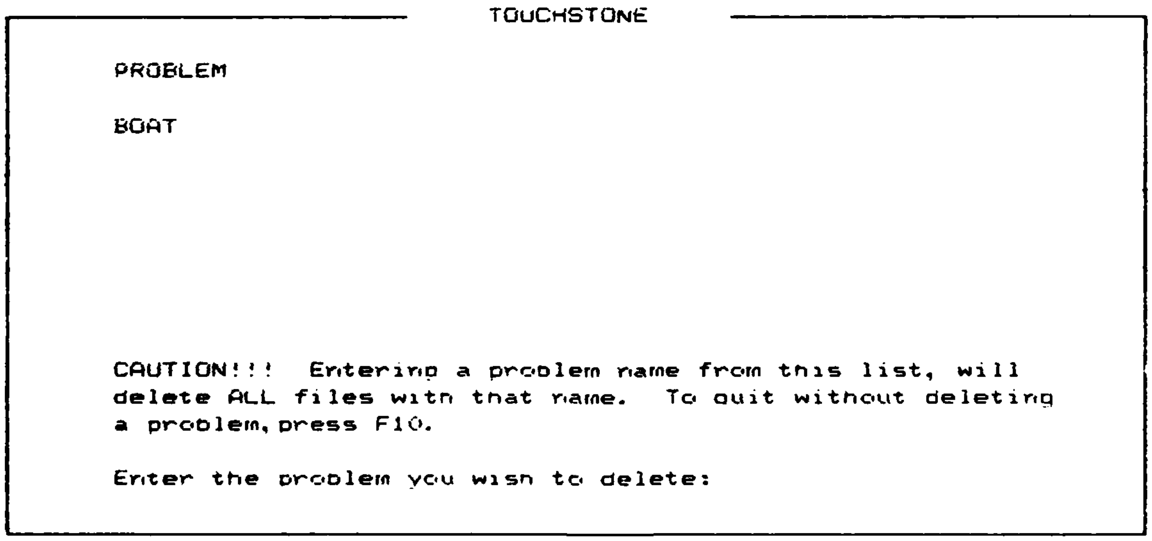
```
                         TOUCHSTONE

    Please enter the name of the new problem.
    The name must not exceed seven letters:           BOAT
    Please give a one line definition of the problem:
    I WOULD LIKE TO BUY A BOAT
    Do you wish to elaborate on that definition?              Y
    ------------------ PROBLEM EXPLANATION ----------------
      This is a chance to buy a boat, but I need to
      know how big, how powerful a boat to buy and
      within what price range I should consider a boat.

    USE:  UP&DN ARROW KEYS.HOME,END,PG UP,PG DN.F-10(quit)
```

FIGURE 21
PROBLEM CREATION SCREEN
(after PROBLEM EXPLANATION INSERT)



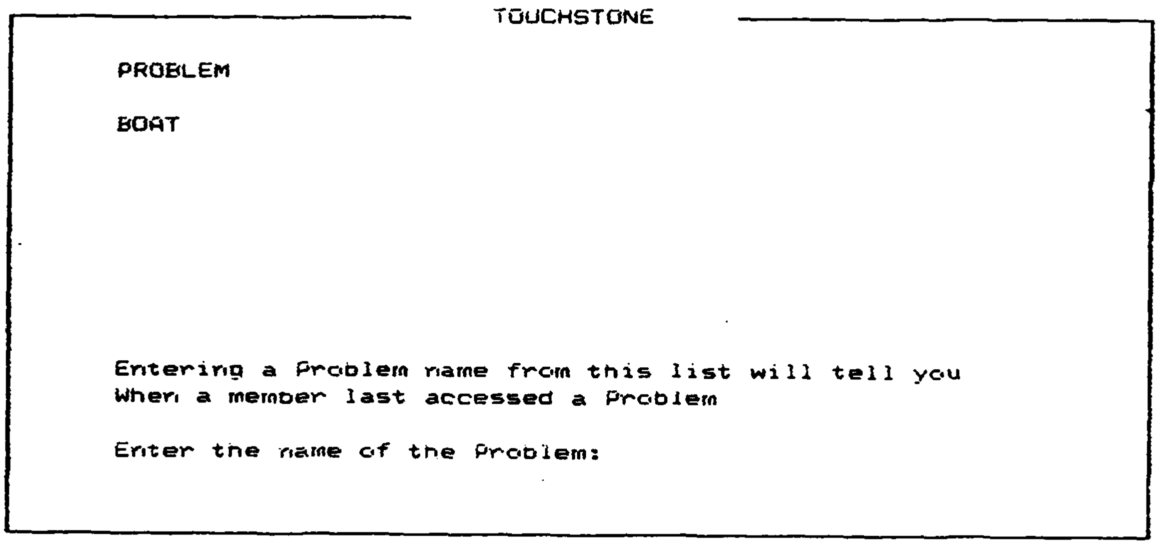
```
                         TOUCHSTONE

    Please enter the name of the new problem.
    The name must not exceed seven letters:          BOAT
    Please give a one line definition of the problem:
    I WOULD LIKE TO BUY A BOAT
    Do you wish to elaborate on that definition?              Y
    How many members comprise this committee?                 2
    Members names:                                          MEN
                                                            BOB
    Will communications and criteria be anonymous?            N
```

FIGURE 22
PROBLEM CREATION SCREEN



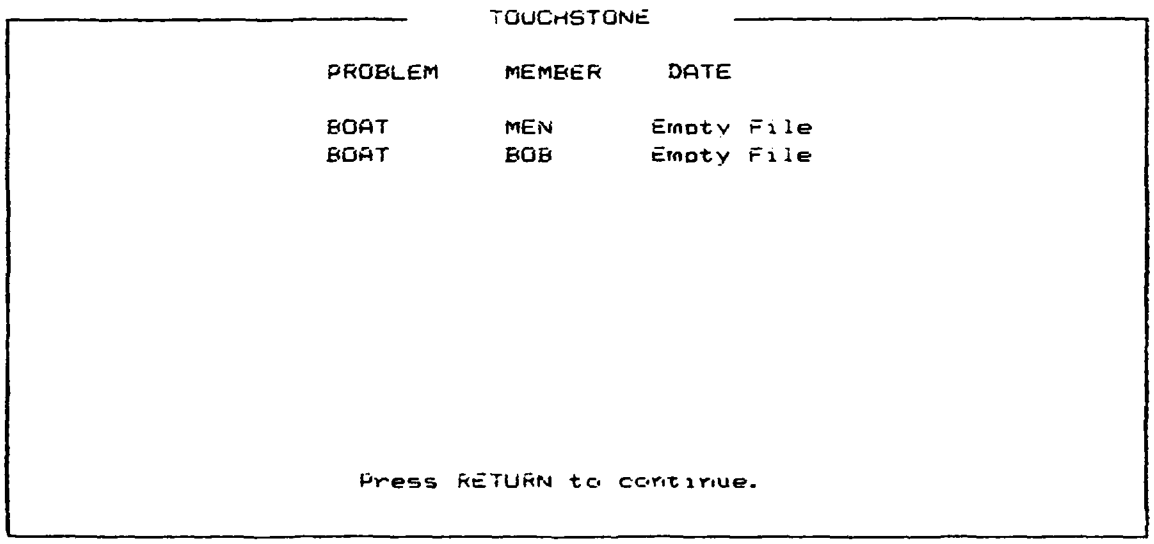
```
                         TOUCHSTONE

   PROBLEM

   BOAT







   CAUTION!!!  Entering a problem name from this list, will
   delete ALL files with that name.  To quit without deleting
   a problem, press F10.

   Enter the problem you wish to delete:
```

FIGURE 23
PROBLEM STATUS CHECK SCREEN (1)



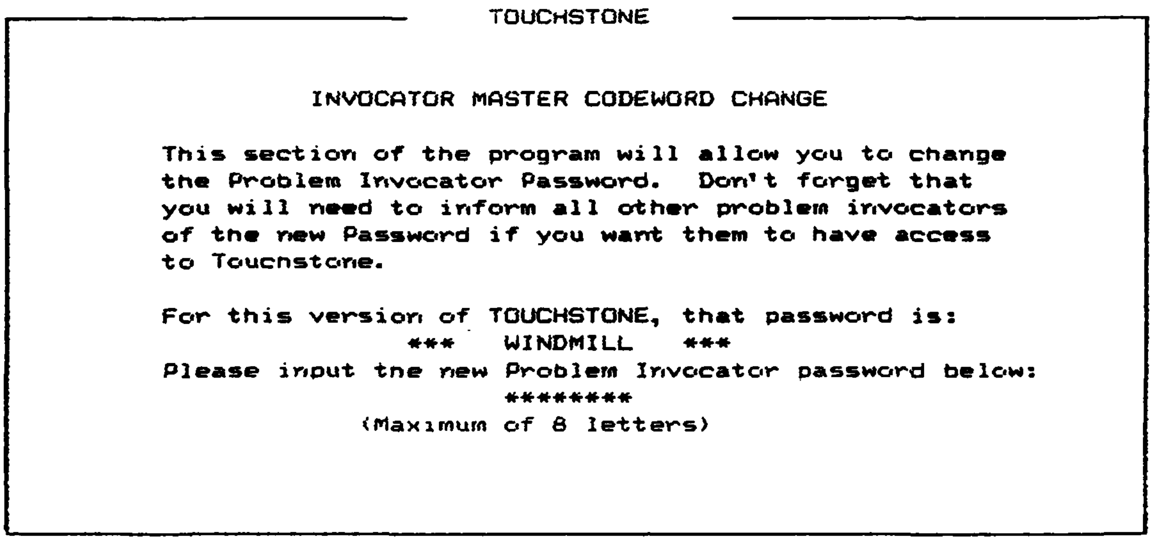
```
                         TOUCHSTONE

    PROBLEM

    BOAT




    Entering a Problem name from this list will tell you
    When a member last accessed a Problem

    Enter the name of the Problem:
```

FIGURE 24
PROBLEM STATUS CHECK SCREEN (2)



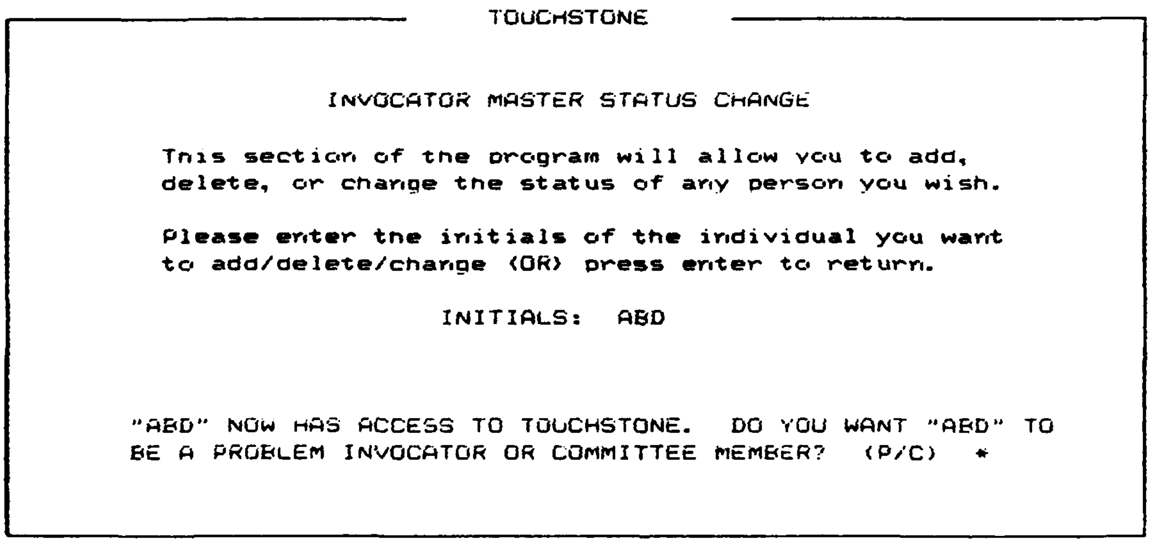
```
                         TOUCHSTONE

             PROBLEM     MEMBER     DATE

             BOAT        MEN       Empty File
             BOAT        BOB       Empty File




                  Press RETURN to continue.
```

FIGURE 25
CHANGE PROBLEM INVOCATOR PASSWORD SCREEN



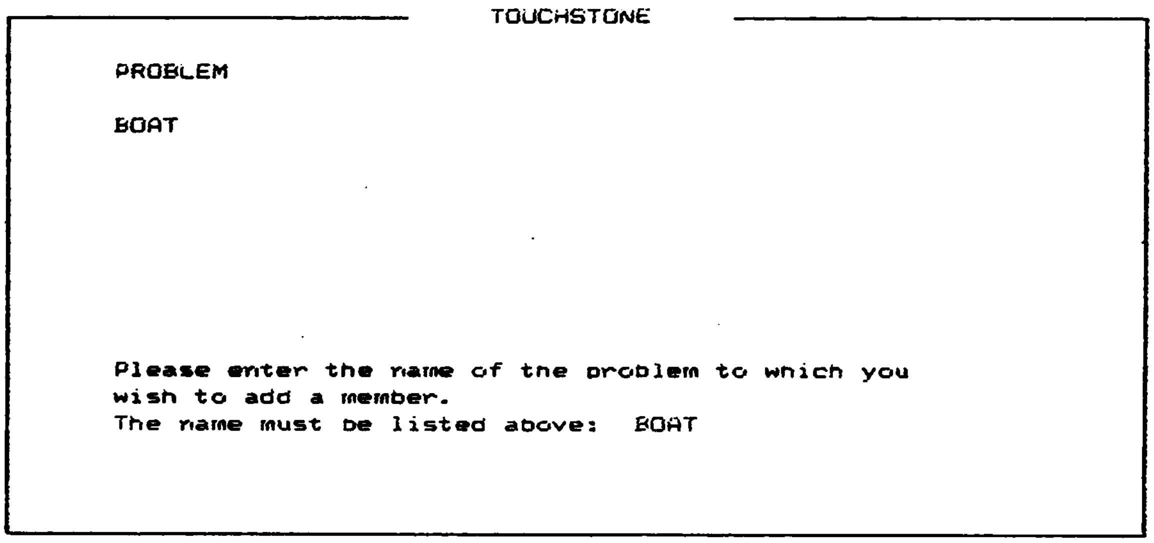
```
                         TOUCHSTONE

               INVOCATOR MASTER CODEWORD CHANGE

        This section of the program will allow you to change
        the Problem Invocator Password.  Don't forget that
        you will need to inform all other problem invocators
        of the new Password if you want them to have access
        to Toucnstone.

        For this version of TOUCHSTONE, that password is:
                       ***   WINDMILL   ***
        Please input tne new Problem Invocator password below:
                              ********
                     (Maximum of 8 letters)
```

FIGURE 26
ADD/DELETE PROBLEM INVOCATOR/COMMITTEE MEMBER SCREEN



```
                         TOUCHSTONE

                INVOCATOR MASTER STATUS CHANGE

        Tnis section of tne program will allow you to add,
        delete, or change the status of any person you wish.

        Please enter tne initials of the individual you want
        to add/delete/change (OR) press enter to return.

                        INITIALS:  ABD


     "ABD" NOW HAS ACCESS TO TOUCHSTONE.  DO YOU WANT "ABD" TO
     BE A PROBLEM INVOCATOR OR COMMITTEE MEMBER?  (P/C)  *
```

FIGURE 27
SCREEN TO ADD A COMMITTEE MEMBER

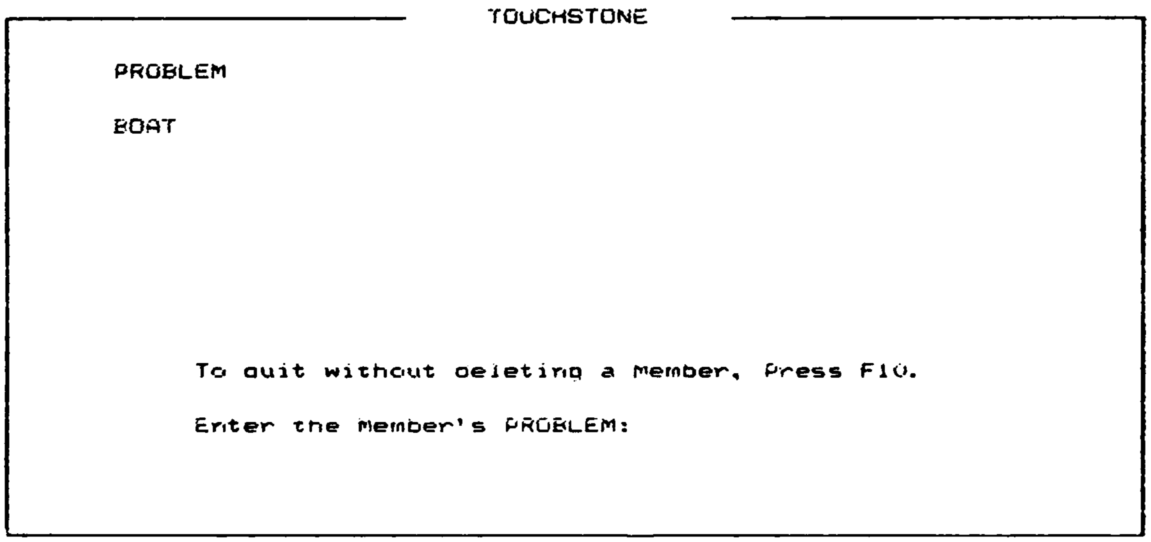

TOUCHSTONE

PROBLEM

BOAT

Please enter the name of the problem to which you
wish to add a member.
The name must be listed above: BOAT

FIGURE 28
SCREEN TO DELETE A COMMITTEE MEMBER
FROM AN EXISTING COMMITTEE


TOUCHSTONE

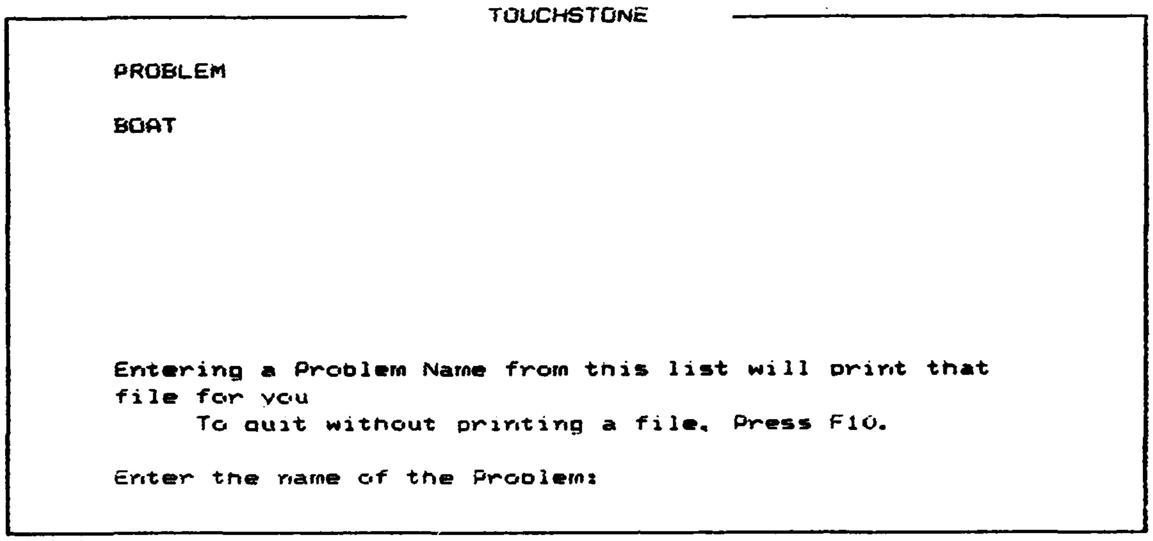
PROBLEM

BOAT

To quit without deleting a member, press F10.

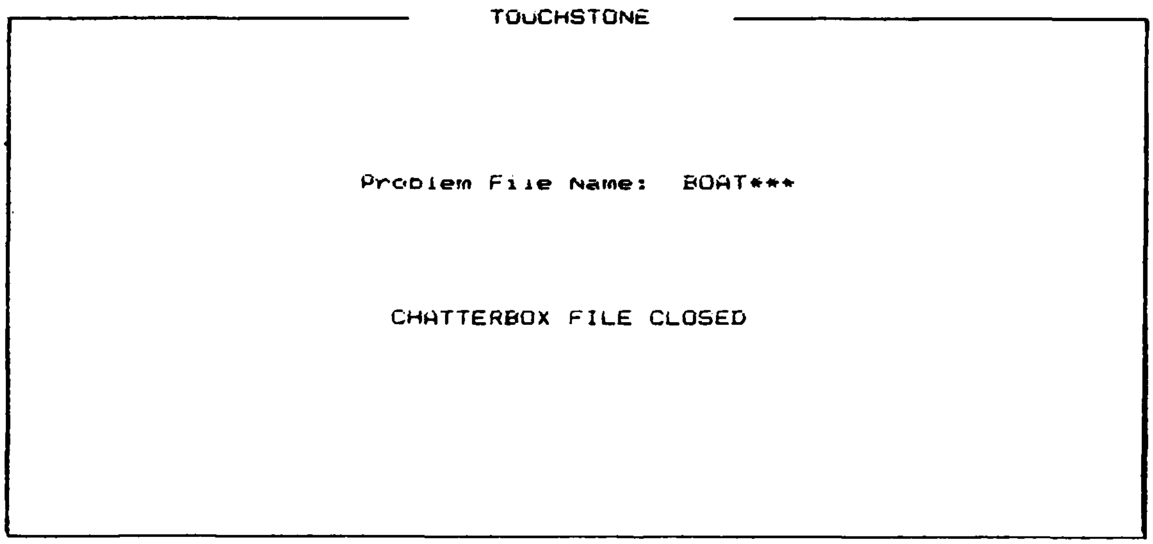
Enter the member's PROBLEM:

FIGURE 29
PRINTOUT SCREEN



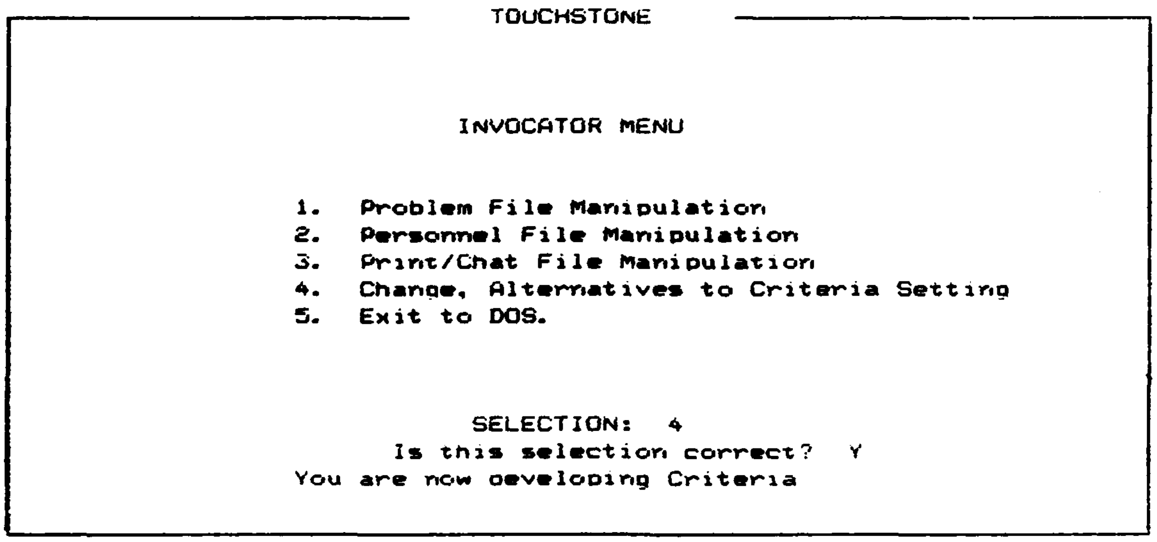
```
                         TOUCHSTONE

    PROBLEM

    BOAT




    Entering a Problem Name from this list will print that
    file for you
         To quit without printing a file, Press F10.

    Enter the name of the Problem:
```

FIGURE 30
CLOSE CHATTERBOX FILE
(IF LEFT OPEN ACCIDENTALLY)



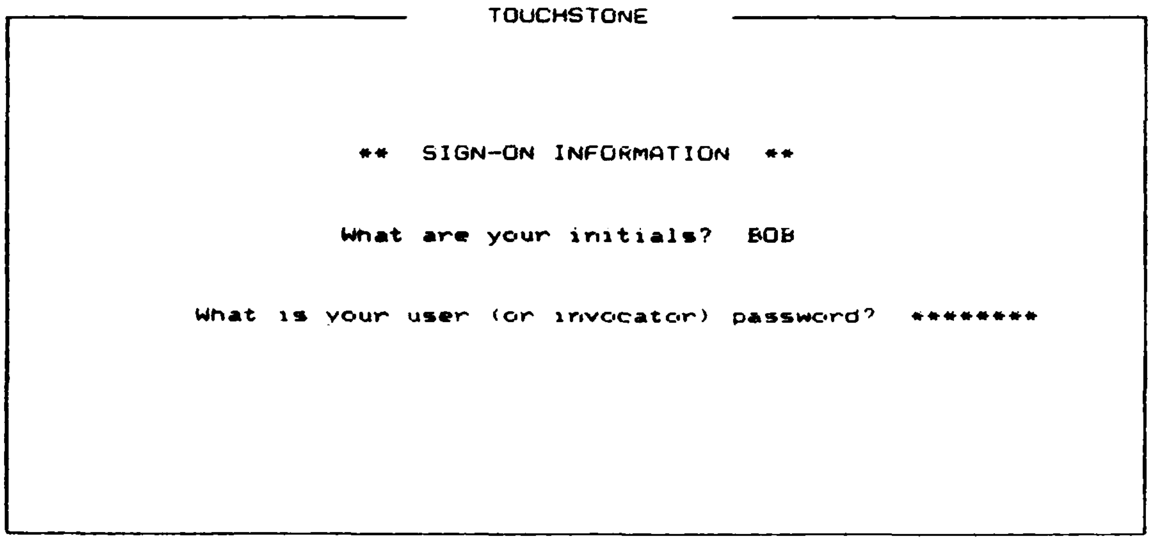
```
                         TOUCHSTONE


              Problem File Name:   BOAT***



                CHATTERBOX FILE CLOSED
```

FIGURE 31
SCREEN SHOWING CHANGE OF ALTERNATIVES TO CRITERIA



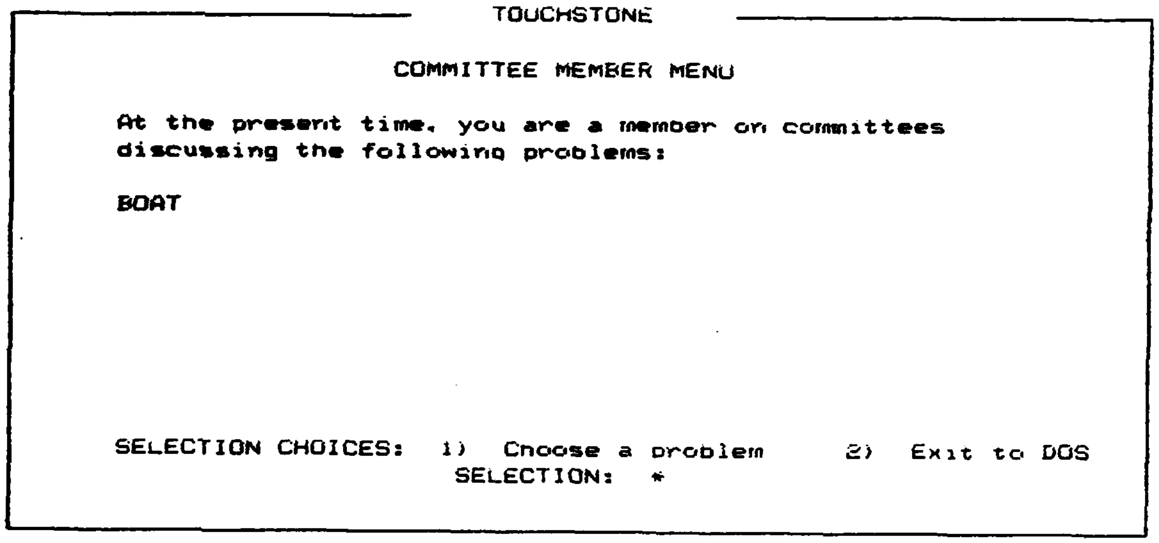
```
                          TOUCHSTONE

                        INVOCATOR MENU

          1.   Problem File Manipulation
          2.   Personnel File Manipulation
          3.   Print/Chat File Manipulation
          4.   Change, Alternatives to Criteria Setting
          5.   Exit to DOS.

                      SELECTION:  4
                Is this selection correct?  Y
          You are now developing Criteria
```

FIGURE 32
COMMITTEE MEMBER SIGN-ON SCREEN



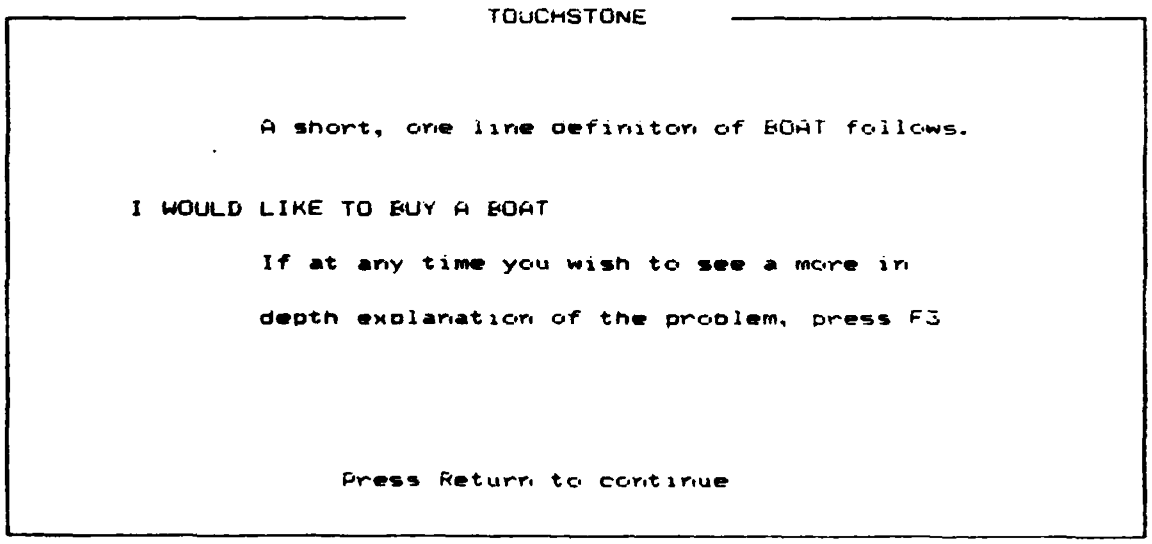
```
                          TOUCHSTONE

                 **  SIGN-ON INFORMATION  **

               What are your initials?  BOB

     What is your user (or invocator) password?  ********
```

FIGURE 33
COMMITTEE MEMBER MENU SCREEN



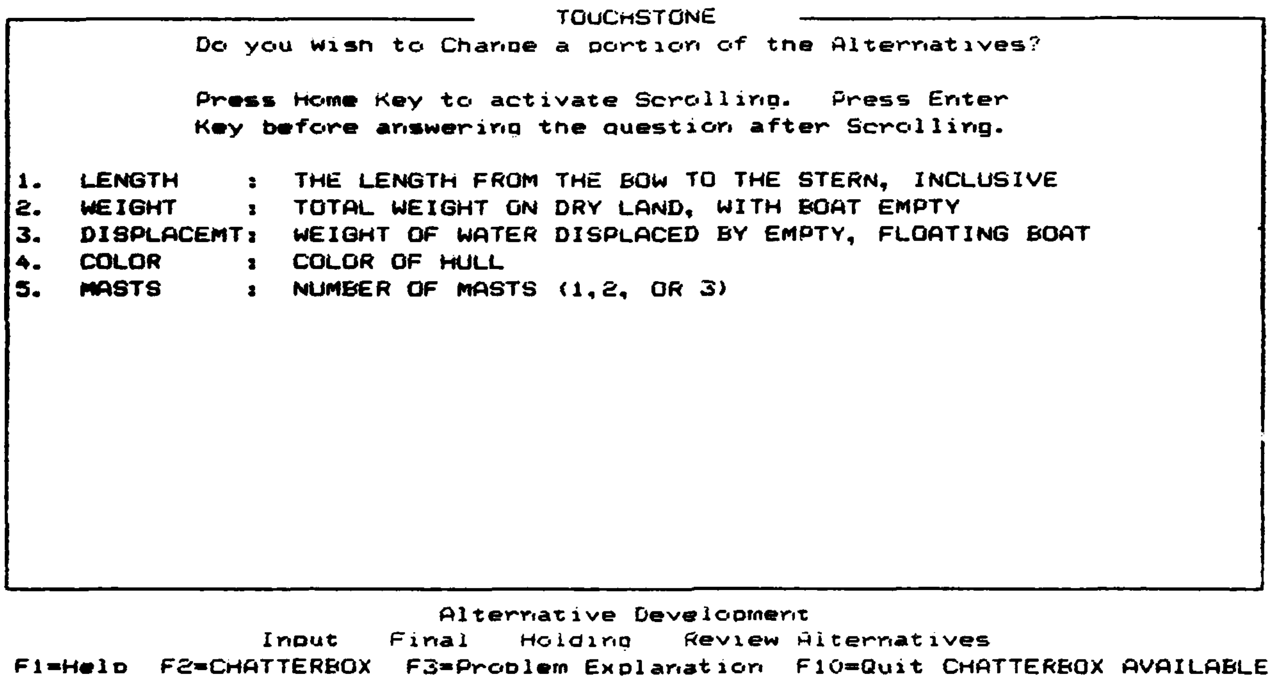
```
                           TOUCHSTONE

                      COMMITTEE MEMBER MENU

    At the present time, you are a member on committees
    discussing the following problems:

    BOAT




    SELECTION CHOICES:  1)  Choose a problem      2)  Exit to DOS
                          SELECTION:  *
```

FIGURE 34
COMMITTEE MEMBER PROBLEM INTRODUCTION SCREEN



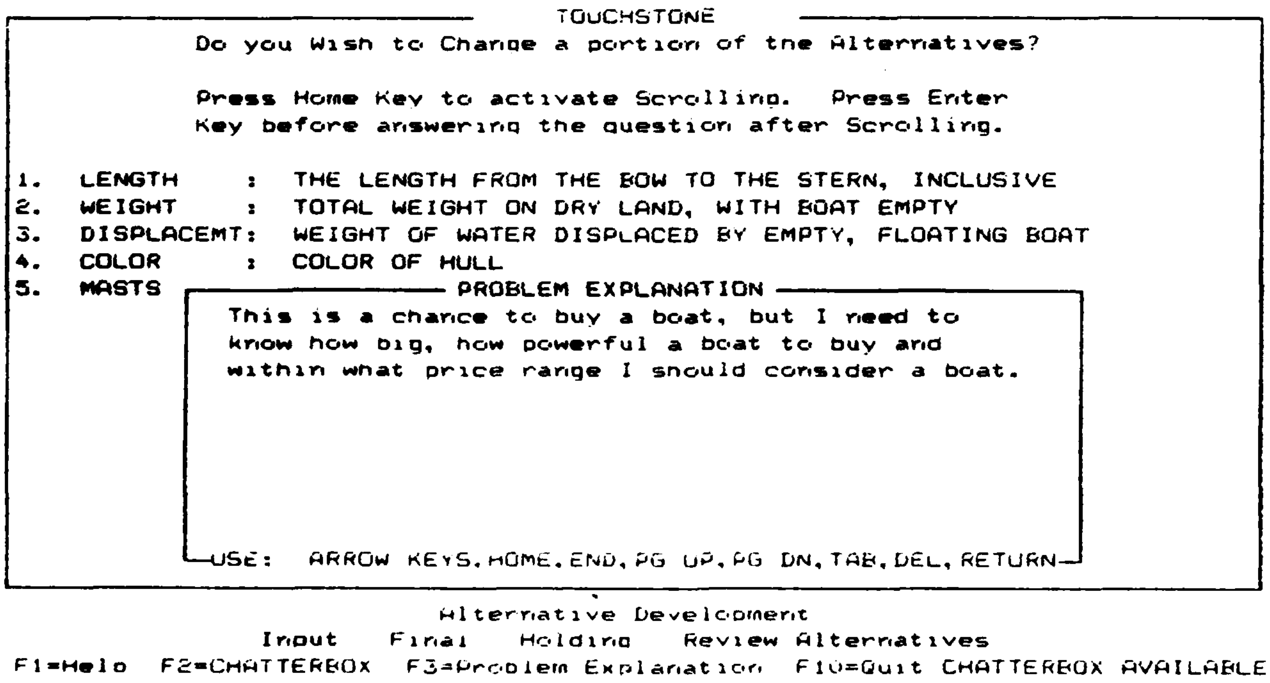
```
                           TOUCHSTONE


           A short, one line definiton of BOAT follows.

     I WOULD LIKE TO BUY A BOAT

           If at any time you wish to see a more in

           depth explanation of the problem, press F3



               Press Return to continue
```

FIGURE 35
SAMPLE COMMITTEE MEMBER WORK SCREEN



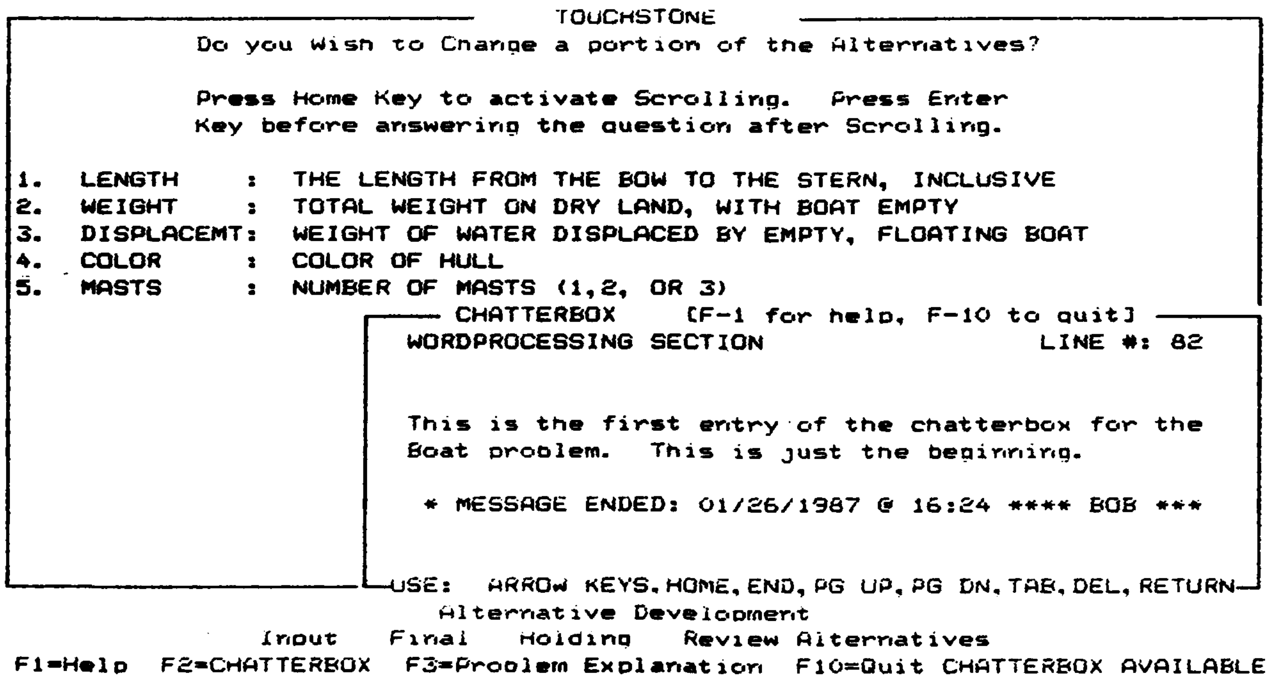
```
                              TOUCHSTONE
            Do you wish to Change a portion of the Alternatives?

            Press Home Key to activate Scrolling.  Press Enter
            Key before answering the question after Scrolling.

1.   LENGTH     :  THE LENGTH FROM THE BOW TO THE STERN, INCLUSIVE
2.   WEIGHT     :  TOTAL WEIGHT ON DRY LAND, WITH BOAT EMPTY
3.   DISPLACEMT:   WEIGHT OF WATER DISPLACED BY EMPTY, FLOATING BOAT
4.   COLOR      :  COLOR OF HULL
5.   MASTS      :  NUMBER OF MASTS (1,2, OR 3)

                      Alternative Development
             Input   Final   Holding   Review Alternatives
F1=Help  F2=CHATTERBOX  F3=Problem Explanation  F10=Quit CHATTERBOX AVAILABLE
```

FIGURE 36
SAMPLE COMMITTEE MEMBER WORK SCREEN
(with PROBLEM EXPLANATION INSERT)



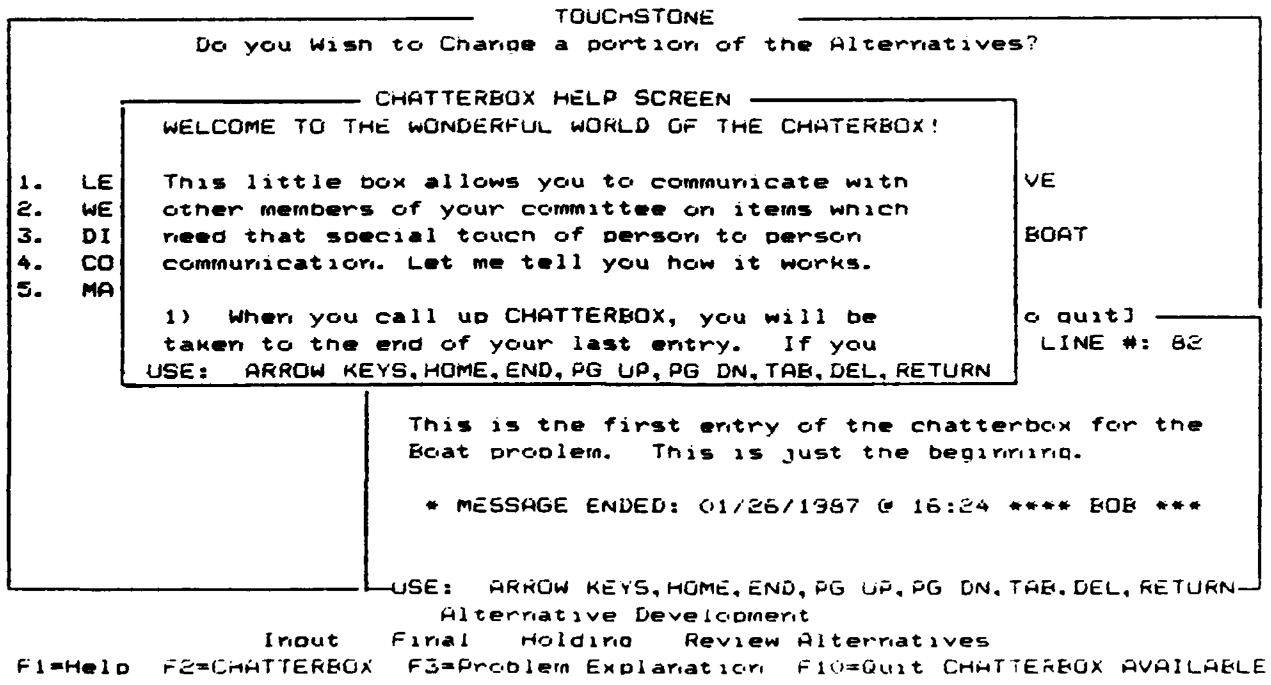
```
                              TOUCHSTONE
            Do you Wish to Change a portion of the Alternatives?

            Press Home Key to activate Scrolling.  Press Enter
            Key before answering the question after Scrolling.

1.   LENGTH     :  THE LENGTH FROM THE BOW TO THE STERN, INCLUSIVE
2.   WEIGHT     :  TOTAL WEIGHT ON DRY LAND, WITH BOAT EMPTY
3.   DISPLACEMT:   WEIGHT OF WATER DISPLACED BY EMPTY, FLOATING BOAT
4.   COLOR      :  COLOR OF HULL
5.   MASTS      ---------- PROBLEM EXPLANATION ----------
              This is a chance to buy a boat, but I need to
              know how big, how powerful a boat to buy and
              within what price range I should consider a boat.

            USE:  ARROW KEYS,HOME,END,PG UP,PG DN,TAB,DEL,RETURN

                      Alternative Development
             Input   Final   Holding   Review Alternatives
F1=Help  F2=CHATTERBOX  F3=Problem Explanation  F10=Quit CHATTERBOX AVAILABLE
```

FIGURE 37
SAMPLE COMMITTEE MEMBER WORK SCREEN
(with CHATTERBOX INSERT)



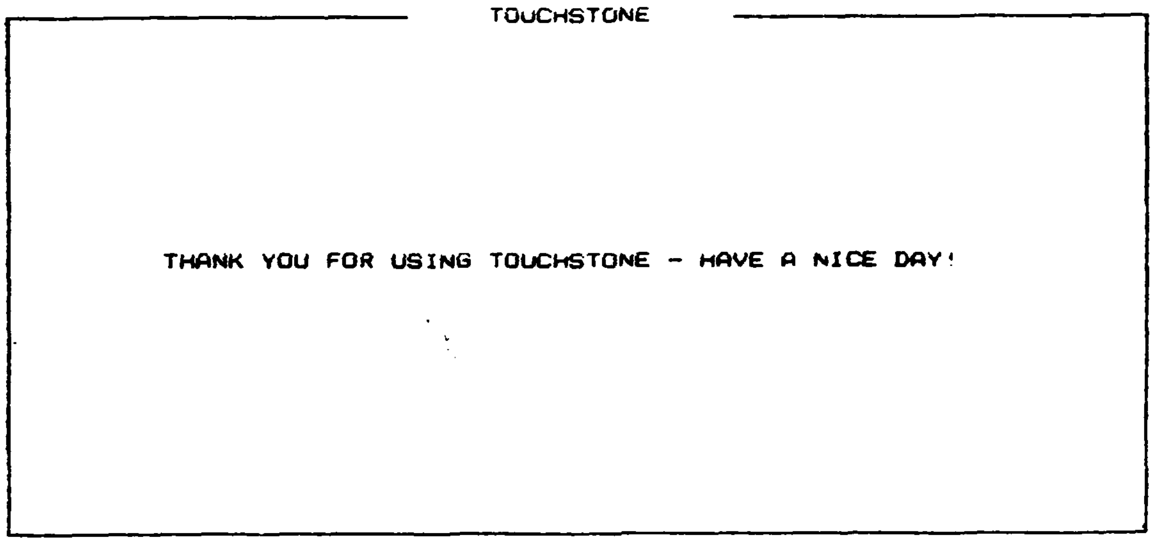
```
------------------------------------ TOUCHSTONE ----------------------------------------
            Do you wish to Change a portion of the Alternatives?

            Press Home Key to activate Scrolling.  Press Enter
            Key before answering the question after Scrolling.

1.  LENGTH     :  THE LENGTH FROM THE BOW TO THE STERN, INCLUSIVE
2.  WEIGHT     :  TOTAL WEIGHT ON DRY LAND, WITH BOAT EMPTY
3.  DISPLACEMT:   WEIGHT OF WATER DISPLACED BY EMPTY, FLOATING BOAT
4.  COLOR      :  COLOR OF HULL
5.  MASTS      :  NUMBER OF MASTS (1,2, OR 3)
                      ----- CHATTERBOX     [F-1 for help, F-10 to quit] -----
                       WORDPROCESSING SECTION                   LINE #: 82


                       This is the first entry of the chatterbox for the
                       Boat problem.  This is just the beginning.

                         * MESSAGE ENDED: 01/26/1987 @ 16:24 **** BOB ***

                      USE:  ARROW KEYS,HOME,END,PG UP,PG DN,TAB,DEL,RETURN
                         Alternative Development
               Input    Final    Holding    Review Alternatives
F1=Help  F2=CHATTERBOX  F3=Problem Explanation  F10=Quit CHATTERBOX AVAILABLE
```

FIGURE 38
SAMPLE COMMITTEE MEMBER WORK SCREEN
(with CHATTERBOX HELP SCREEN INSERT)



```
------------------------------------ TOUCHSTONE ----------------------------------------
            Do you wish to Change a portion of the Alternatives?

         ----------------- CHATTERBOX HELP SCREEN -------------
          WELCOME TO THE WONDERFUL WORLD OF THE CHATERBOX!

1.  LE    This little box allows you to communicate with      VE
2.  WE    other members of your committee on items which
3.  DI    need that special touch of person to person         BOAT
4.  CO    communication. Let me tell you how it works.
5.  MA
          1)  When you call up CHATTERBOX, you will be        o quit] ---
          taken to the end of your last entry.  If you         LINE #: 82
         USE:  ARROW KEYS,HOME,END,PG UP,PG DN,TAB,DEL,RETURN

                       This is the first entry of the chatterbox for the
                       Boat problem.  This is just the beginning.

                         * MESSAGE ENDED: 01/26/1987 @ 16:24 **** BOB ***

                      USE:  ARROW KEYS,HOME,END,PG UP,PG DN,TAB,DEL,RETURN
                         Alternative Development
               Input    Final    Holding    Review Alternatives
F1=Help  F2=CHATTERBOX  F3=Problem Explanation  F10=Quit CHATTERBOX AVAILABLE
```

FIGURE 39
SIGN-OFF SCREEN


TOUCHSTONE

THANK YOU FOR USING TOUCHSTONE - HAVE A NICE DAY!

APPENDIX D
PROGRAM LISTING

```
program ATOUCH;

type
  CODEARRAY                                   = string[12];

var
  HELPDRIVE, FILEDRIVE, AUTHORITY             : char;
  INVOCATOR                                   : char;
  TEMPFILE                                    : text;
  NAMESTRING, NAMECHECK                       : string[3];
  USERCODE                                    : string[8];
  TEMPNAME, CODENAME                          : string[12];

{$IFILTERA.LIB}
{$IFILTERB.LIB}
{$IFILTERC.LIB}

begin
  TITLE;
  GETTHEDATE;
  INTRODUCTION;
  clrscr;
  gotoxy (14,8);
  write ('Checking files - please stand by');
  CHECKTHEFILES;
  gotoxy (14,8);
  write ('Checking files - please stand by');
  INVOCATOR := 'W';
  VERIFYCODE;
  assign (TEMPFILE, 'DRIVEFIL.TMP');
  rewrite (TEMPFILE);
  TEMPNAME := concat(HELPDRIVE, FILEDRIVE, AUTHORITY,
                     NAMESTRING, INVOCATOR,'KIMMY');
  CODENAME := ENCODE (TEMPNAME);
  writeln (TEMPFILE,CODENAME);
  close(TEMPFILE);
end.    {program TOUCHSTA}
```

```
program BTOUCH(INPUT,OUTPUT);


type
   STRING1                  = STRING[1];
   STRING3                  = STRING[3];
   STRING8                  = STRING[8];
   STRING10                 = STRING[10];
   STRING12                 = STRING[12];

   PROBREC                  = record
                                 .CHECKSTATE       : CHAR;
                                  CHECKCHANGE      : CHAR;
                                  CHOICE           : CHAR;
                                  PROBLEM          : STRING[7];
                                  NUMMEMS          : INTEGER;
                                  MEMBER           : STRING3;
                                  DEFINITION       : STRING[58];
                                  DATELINE         : STRING12;
                              end;

   CRIREC                   = record
                                  FLAG1         : INTEGER;
                                  FLAG2         : INTEGER;
                                  FLAG3         : INTEGER;
                                  CHECKPOINT    : INTEGER;
                                  STATFLAG      : CHAR;
                                  CRITNAME      : STRING10;
                                  CRITDEF       : STRING[58];
                              end;

   CODEARRAY                    = STRING12;
   STRINGARRAY                  = array[1..59] of CHAR;
   CRITARRAY                    = array[1..50] of CRIREC;
   PROBARRAY                    = array[1..200] of PROBREC;


var

   HELPDRIVE, CHT,
   INVOCATOR, CHANGEREC            : CHAR;
   FILEDRIVE,
   PROBLEMFLAG, HELPER, ALT        :  CHAR;

   STOPGAP, CHATOK,   SCROLLIT,
   WEEDDEF, FILECHECK              :  BOOLEAN;
   ANONYMOUS, STARTUP, STOPPROG,
   AUTHORIZED, PRINTONE            :  BOOLEAN;

   I,   J,   PT1,   COUNT,
   COUNTED,    MOVEX, M          :  INTEGER;
   W,   X,   PT2,   LIMMIT,
   CLEARIT,    MOVEOVER          :  INTEGER;
   Y,   N,   PT3,   TRACK1,
   COUNTER,                             :  INTEGER;
   Z,    A,    PT4,   SECNUM,
   SELECTED,   FLAGCOUNT                :  INTEGER;
   B,    L,    NUM,   THRNUM,
   HELPSIZE,                            :  INTEGER;


   CH, CHA, NEWPROB, CHOICE,
   ALTERNATIVE                           :  STRING1;
   NAMESTRING, NEWNAME                   :  STRING3;
   PROBNAME                              :  STRING[7];
   NEWSTRING, CHATRFILE, DATE            :  STRING12;

   PROBS                :  PROBARRAY;
   NAMES                :  CRITARRAY;
   INPUTSTRING          :  STRINGARRAY;

   CRITERIA             :  CRIREC;
   MEMBERS              :  PROBREC;

   KRITERIAFILE         :  file of CRIREC;
   ACTIVEPROBLEMFILE    :  file of PROBREC;

{$IFILTER1.LIB}
{$IFILTER2.LIB}
{$IFILTER3.LIB}
{$IFILTER4.LIB}
{$IFILTER7.LIB}
{$IFILTER9.LIB}
{$IFRONTEND.LIB}


procedure ProbManipulation;

(**********************************************************
 *  PROCEDURE            :  PROBMANIPULATION              *
 *  SUPPORTS PROGRAM     :  8TOUCH.PAS                    *
 *  LOCAL VARIABLES      :  CH, SELECTION, CONTINUE,      *
 *                          COMPLETED, CODE               *
 *  GLOBAL VARIABLES     :  INPUTSTRING, SELECTED         *
 *  ARRAYS USED          :  NONE                          *
 *  EXTERNAL CALLS       :  NEWPROBLEM, DELETEAPROBLEM,   *
 *                          CHECKAPROBLEM,                *
 *                          LOADEMUP, GETTHEKEYS          *
 *  EXTERNAL FILTERS     :  FILTER9.LIB, FRONTEND.LIB     *
 *  CALLED FROM          :  WINDOW1                       *
 *  PURPOSE              :  SETS UP A MENU SCREEN FOR THE *
 *                          INVOCATOR TO ACCESS THREE     *
 *                          DIFFERENT MANIPULATIONS       *
 *                          CONCERNING PROBLEMS.          *
 *********************************************************

var
   CH                         : char;
```

```
    SELECTION                   : STRING[1];
    CONTINUE,COMPLETED          : boolean;
    TEMPFLAGSET                 : TEXT;
    CODE                        : INTEGER;

    begin   {probmanipulation}
       completed := false;
       repeat
          clrscr;
          gotoxy(22,3);     write ('INVOCATOR MENU');
          gotoxy(14,7);     write ('1.  Begin New Problem.');
          gotoxy(14,8);     write ('2.  Delete a Problem.');
          gotoxy(14,9);     write ('3.  Check Status on a
Specific Problem.');
          gotoxy(14,10);    write ('4.  Exit to Main Menu');
          gotoxy(23,14);    write ('SELECTION:   *');
          repeat
             gotoxy (35,14);             write ('*');
             repeat
                gotoxy (35,14);
                getthekeys(inputstring,1);
                SELECTION := inputstring;
                val(Selection,Selected,code);
             until SELECTED in [1..4];
             gotoxy (12,15);       write('Is this selection
correct?  Y');
             gotoxy (40,15);       write ('Y');
             gotoxy (40,15);
             repeat
                read (kbd,CH);
                if CH in ['y','n'] then
                   CH := chr(ord(CH)-32);
             until CH in ['Y','N',#13];
             write(CH);
             if CH in ['N'] then
                CONTINUE := false
             else
                CONTINUE := true;
          until CONTINUE;
          case SELECTED of
             1 : NewProblem;
             2 : DeleteAProblem;
             3 : CheckAProblem;
             4 : COMPLETED := true;
          end;  {case CH}
          LoadEmUp;
       until completed;
       completed := false;
    end;    {probmanipulation}


procedure PersManipulation;

{**************************************************************
 *  PROCEDURE           :  PROCDURE PERSMANIPULATION          *
```

```
  *  SUPPORTS PROGRAM    :  BTOUCH.PAS                                *
  *  LOCAL VARIABLES     :  CH, SELECTION, CONTINUE,                  *
  *                        COMPLETED, CODE                           *
  *  GLOBAL VARIABLES    :  INPUTSTRING, SELECTED                     *
  *  ARRAYS USED         :  NONE                                      *
  *  FILES ACCESSED      :  NONE                                      *
  *  EXTERNAL CALLS      :  GETTHEKEYS, CHANGESTATUS.                 *
  *                         ADDAMEMBER,   DELETEAMEMBER,              *
  *                         LOADEMUP                                  *
  *  EXTERNAL FILTERS    :  FILTER7.LIB, FILTER9.LIB,                 *
  *                         FRONTEND.LIB                              *
  *  CALLED FROM         :  WINDOW1                                   *
  *  PURPOSE             :  SETS UP A MENU SCREEN FOR THE             *
  *                         INVOCATOR TO ACCESS THREE                 *
  *                         DIFFERENT MANIPULATIONS                   *
  *                         CONCERNING PERSONNEL.                     *
  ***************************************************************)

var
   CH                          : char;
   SELECTION                   : STRING[1];
   CONTINUE,COMPLETED          : boolean;
   CODE                        : INTEGER;

   begin   {PersManipulation}
      completed := false;
      repeat
         clrscr;
         gotoxy(22,3);   write ('INVOCATOR MENU');
         gotoxy(6,6);   write ('1.  Change Problem Invocator
Password.');
         gotoxy(6,7);   write ('2.  Add/Delete a Problem
Invocator.');
         gotoxy(6,8);
         write ('3.  Add a Committee Member To An Existing
Committee.');
         gotoxy(6,9);
         write ('4.  Delete a Member From An Existing
Committee.');
         gotoxy(6,10);   write ('5.  Exit to Main Menu');
         gotoxy(23,14);  write ('SELECTION:  *');
         repeat
            gotoxy (35,14);            write ('*');
            repeat
               gotoxy (35,14);
               getthekeys(inputstring,1);
               SELECTION := inputstring;
               val(Selection,Selected,code);
            until SELECTED in [1..5];
            gotoxy (12,15);       write('Is this selection
correct?  Y');
            gotoxy (40,15);       write ('Y');
            gotoxy (40,15);
            repeat
               read (kbd,CH);
```

```
                if CH in ['y','n'] then
                   CH := chr(ord(CH)-32);
             until CH in ['Y','N',#13];
             write(CH);
             if CH in ['N'] then
                CONTINUE := false
             else
                CONTINUE := true;
          until CONTINUE;
          case SELECTED of
             1 : CHANGESTATUS;
             2 : changestatus;
             3 : AddAMember;
             4 : DeleteAMember;
             5 : COMPLETED := true;
          end;  {case CH}
          LoadEmUp;
       until completed;
       completed := false;
    end;    {PersManipulation}


procedure ChatManipulation;

(***********************************************************
 *  PROCEDURE           :  CHATMANIPULATION                *
 *  SUPPORTS PROGRAM    :  BTOUCH.PAS                      *
 *  LOCAL VARIABLES     :  CH, SELECTION, CONTINUE,        *
 *                         COMPLETED, CODE                 *
 *  GLOBAL VARIABLES    :  ALT, SELECTED, INPUTSTRING      *
 *  ARRAYS USED         :  NONE                            *
 *  FILES ACCESSED      :  NONE                            *
 *  EXTERNAL CALLS      :  PRINTCHATTERBOX,                *
 *                         PRINTALTERNATIVES, CLOSEFILE,   *
 *                         LOADEMUP, GETTHEKEYS            *
 *  EXTERNAL FILTERS    :  FRONTEND.LIB                    *
 *  CALLED FROM         :  WINDOW1                         *
 *  PURPOSE             :  SETS UP A MENU SCREEN FOR THE   *
 *                         INOVCATOR TO ACCESS DIFFERENT   *
 *                         MANIPULATIONS CONCERNING THE    *
 *                         PRINTING OF FILES AND CLOSING OF *
 *                         A CHATTERBOX ACCIDENTLY LEFT    *
 *                         OPEN.                           *
 **********************************************************)

var
   CH                          : char;
   SELECTION                   : STRING[1];
   CONTINUE,COMPLETED          : boolean;
   TEMPFLAGSET                 : TEXT;
   CODE                        : INTEGER;

   begin    {ChatManipulation}
      completed := false;
      repeat
```

```
        clrscr;
        gotoxy(22,3);   write ('INVOCATOR MENU');
        gotoxy(8,5);    write ('1.  Print Out Chatterbox
for Alternatives.');
        gotoxy(8,6);    write ('2.  Print Out Chatterbox
for Criteria.');
        gotoxy(8,7);    write ('3.  Close a Chatterbox File
Which Has Been');
        gotoxy(8,8);    write ('     Left Open
Accidentally.');
        gotoxy(8,9);    write ('4.  Print Out Developed
Alternatives.');
        gotoxy(8,10);    write ('5.  Print Out Developed
Criteria.');
        gotoxy(8,11);   write ('6.  Exit to Main Menu');
        gotoxy(23,14);  write ('SELECTION:  *');
        repeat
           gotoxy (35,14);            write ('*');
           repeat
              gotoxy (35,14);
              getthekeys(inputstring,1);
              SELECTION := inputstring;
              val(Selection,Selected,code);
           until SELECTED in [1..6];
           gotoxy (12,15);       write('Is this selection
correct?  Y');
           gotoxy (40,15);       write ('Y');
           gotoxy (40,15);
           repeat
              read (kbd,CH);
              if CH in ['y','n'] then
                 CH := chr(ord(CH)-32);
           until CH in ['Y','N',#13];
           write(CH);
           if CH in ['N'] then
              CONTINUE := false
           else
              CONTINUE := true;
        until CONTINUE;
        case SELECTED of
           1 : begin
                  alt := 'A';
                  printchatterbox;
               end;
           2 : begin
                  alt := 'C';
                  printchatterbox;
               end;
           3 : closefile;
           4 : begin
                  alt := 'A';
                  printalternatives;
               end;
           5 : begin
                  alt := 'C';
```

```
                    printalternatives;
                 end;
             6 : COMPLETED := true;
          end;  {case CH}
          LoadEmUp;
       until completed;
       completed := false;
    end;    {ChatManipulation}


procedure Window1;

(*************************************************************
 *  PROCEDURE           :  WINDOW1                           *
 *  SUPPORTS PROGRAM    :  BTOUCH.PAS                        *
 *  LOCAL VARIABLES     :  CH, SELECTION, CONTINUE,          *
 *                         COMPLETED, CODE, TEMPALT,         *
 *                         TEMPALTER                         *
 *  GLOBAL VARIABLES    :  HELPER, HELPSIZE. ALTERNATIVE,    *
 *                         INPUTSTRING, SELECTED, CHATOK,    *
 *                         NAMESTRING, FILEDRIVE             *
 *  ARRAYS USED         :  NONE                              *
 *  FILES ACCESSED      :  ACTIVEPROBLEMFILE                 *
 *  EXTERNAL CALLS      :  INTROSCREEN, PROBMANIPULATION.    *
 *                         PERSMANIPULATION                  *
 *                         CHATMANIPULATION, LOADEMUP        *
 *  EXTERNAL FILTERS    :  FILTER9.LIB                       *
 *  CALLED FROM         :  MAIN BODY OF PROGRAM BTOUCH.PAS   *
 *  PURPOSE             :  THIS PROCEDURE PROVIDES THE MAIN  *
 *                         SCREEN THE INVOCATOR WORKS FROM.  *
 *                         HE WILL ACCESS ALL OTHER          *
 *                         INVOCATOR ACTIVITIES FROM THIS    *
 *                         PROCEDURE. AND EXIT TO DOS WHEN   *
 *                         THESE ACTIONS ARE COMPLETED.      *
 *************************************************************)

var
   CH                         : char;
   SELECTION                  : STRING[1];
   CONTINUE,COMPLETED         : boolean;
   TEMPFLAGSET                : TEXT;
   CODE                       : INTEGER;
   TEMPALT, TEMPALTER         : STRING[12];

   begin   {Window1}
      COMPLETED := false;
      repeat
         Assign(activeproblemfile,concat(filedrive,
':probs.txt'));
         INTROSCREEN;
         HELPER := 'C';
         HELPSIZE := 100;
         if alternative = 'A' then
            begin
               tempalt := 'Alternatives';
```

```
                    tempalter := 'Criteria';
                 end
              else
                 begin
                    tempalt := 'Criteria';
                    tempalter := 'Alternatives';
                 end;
              gotoxy(22,3);  write ('INVOCATOR MENU');
              gotoxy(12,6);   write ('1.  Problem File
Manipulation');
              gotoxy(12,7);   write ('2.  Personnel File
Manipulation');
              gotoxy(12,8);   write ('3.  Print/Chat File
Manipulation');
              gotoxy(12,9);   write ('4.  Change, ', tempalt,' to
',tempalter,
                                          ' Setting');
              gotoxy(12,10);  write ('5.  Exit to DOS.');
              gotoxy(23,14);        write ('SELECTION:  *');
              repeat
                 gotoxy (35,14);            write ('*');
                 repeat
                    gotoxy (35,14);
                    getthekeys(inputstring,1);
                    SELECTION := inputstring;
                    val(Selection,Selected,code);
                 until SELECTED in [1..5];
                 gotoxy (18,15);      write('Is this selection
correct?  Y');
                 gotoxy (46,15);      write ('Y');
                 gotoxy (46,15);
                 repeat
                    read (kbd,CH);
                    if CH in ['y','n'] then
                       CH := chr(ord(CH)-32);
                 until CH in ['Y','N',#13];
                 write(CH);
                 if CH in ['N'] then
                    CONTINUE := false
                 else
                    CONTINUE := true;
              until CONTINUE;
              case SELECTED of
                 1 : ProbManipulation;
                 2 : PersManipulation;
                 3 : ChatManipulation;
                 4 : begin
                        if alternative = 'C' then
                           begin
                              alternative := 'A';
                              tempalt := 'Alternatives';
                           end
                        else
                           begin
                              alternative := 'C';
                              tempalt := 'Criteria';
                           end;
                        gotoxy(12,16);
                        write('You are now developing  ',tempalt);
                     end;
                 5 : COMPLETED := true;
              end;  {case CH}
                LoadEmUp;
           until COMPLETED;
           ChatOK := False;
        end;    {Window1}


     begin    {Main Program}

       INVOCATOR := 'W';
       GETFILENAMES;
       INTROSCREEN;

       if not authorized then begin
         gotoxy(9,9);
         write('ACCESS DENIED - TOUCHSTONE PROGRAM EXITED!');
         delay(2000);
       end;   {if not authorized}

       if (AUTHORIZED) and (invocator = 'M') then begin
         gotoxy(10,8);
         write('ACCESS APPROVED - WELCOME TO TOUCHSTONE!');
         delay(3000);
         ALTERNATECHOICE;

          (***** call touchstone programs *****)
              if INVOCATOR = 'M' then
                window1;
          (*************************************)

         clrscr;
         gotoxy (4,8);
         write ('THANK YOU FOR USING TOUCHSTONE - HAVE A NICE
     DAY!');
         delay (2000);

         authorized := false;

       end;   {if AUTHORIZED}

     end.     {Main Program}
```

```
program CTOUCH(INPUT,OUTPUT);


type
   STRING1                   = STRING[1];
   STRING3                   = STRING[3];
   STRING8                   = STRING[8];
   STRING10                  = STRING[10];
   STRING12                  = STRING[12];

   PROBREC                   = record
                                  CHECKSTATE      : CHAR;
                                  CHECKCHANGE     : CHAR;
                                  CHOICE          : CHAR;
                                  PROBLEM         : STRING[7];
                                  NUMMEMS         : INTEGER;
                                  MEMBER          : STRING3;
                                  DEFINITION      : STRING[58];
                                  DATELINE        : STRING12;
                               end;

   CRIREC                    = record
                                  FLAG1        : INTEGER;
                                  FLAG2        : INTEGER;
                                  FLAG3        : INTEGER;
                                  CHECKPOINT   : INTEGER;
                                  STATFLAG     : CHAR;
                                  CRITNAME     : STRING10;
                                  CRITDEF      : STRING[58];
                               end;

   CODEARRAY                    = STRING12;
   STRINGARRAY                  = array[1..59] of CHAR;
   CRITARRAY                    = array[1..150] of CRIREC;
   PROBARRAY                    = array[1..200] of PROBREC;


var

   HELPDRIVE, CHT,
   INVOCATOR, CHANGEREC        :  CHAR;
   FILEDRIVE,
   PROBLEMFLAG, HELPER, ALT    :  CHAR;

   STOPGAP, CHATOK, SCROLLIT,
   WEEDDEF,  FILECHECK                 :  BOOLEAN;
   ANONYMOUS, STARTUP,
   LINEMARK, STOPPROG, AUTHORIZED  :  BOOLEAN;

   A, QUITFLG1, TRACK1, COUNT,
   HELPSIZE,  PT1,  W                :  INTEGER;
   B, QUITFLG2, MOVEX,
   PT2,  X                           :  INTEGER;
```

```
    I, QUITFLG3, THRNUM, COUNTER,
    MOVEOVER,  PT3,  Y                 :  INTEGER;
    J, CHKFLAG1, LIMMIT, RECOUNT,
    SELECTED,  PT4,  Z                 :  INTEGER;
    L, CHKFLAG2, SECNUM, COUNTED,
    NUM                                :  INTEGER;
    M, CHKFLAG3, SHOWME, CLEARIT,
     CRITLIMIT                         :  INTEGER;
    N, QUITFLAG, MARKER,
    NEWCRITLIMIT                       :  INTEGER;

    FLAGCHOICE, CH, CHA, NEWPROB,
    CHOICE, ALTERNATIVE                :  STRING1;
    NAMESTRING, NEWNAME                :  STRING3;
    PROBNAME                           :  STRING[7];
    NEWSTRING, CHATRFILE, DATE         :  STRING12;

    TEMPFILE                           :  TEXT;

    PROBS                              :  PROBARRAY;
    NAMES                              :  CRITARRAY;
    INPUTSTRING                        :  STRINGARRAY;

    CRITERIA                           :  CRIREC;
    MEMBERS                            :  PROBREC;

    KRITERIAFILE                       :  file of CRIREC;
    ACTIVEPROBLEMFILE                  :  file of PROBREC;

{$IFILTER1.LIB}
{$IFILTER2.LIB}
{$IFILTER3.LIB}
{$IFILTER4.LIB}
{$IFILTER6.LIB}
{$IFILTER7.LIB}
{$IFILTER9.LIB}
{$ITAILEND.LIB}
```

```
procedure InitVariables;

(***********************************************************
 *  PROCEDURE          :  INITVARIABLES                    *
 *  SUPPORTS PROGRAM   :  CTOUCH.PAS                       *
 *  LOCAL VARIABLES    :  CH, TEMPALT                      *
 *  GLOBAL VARIABLES   :  PT1, PT2, PT3, PT4, QUITFLG1,    *
 *                        QUITFLAG2, L, M, N, QUITFLG3,    *
 *                        SHOWME, THRNUM, SECNUM, QUITFLAG, *
 *                        CHKFLAG1, STARTUP, STOPGAP,      *
 *                        SCROLLIT, Y, NUM, CRITLIMIT,     *
 *                        NEWCRITLIMIT, RECOUNT, CHANGEREC, *
 *                        CHA, COUNT, FILEDRIVE, NAMESTRING,*
 *                        PROBNAME, ALTERNATIVE, NEWSTRING, *
 *                        MEMBERS, Z, CRITERIA, PROBLEMFLAG,*
 *                        INPUTSTRING, CHM                 *
 *  ARRAYS USED        :  NONE                             *
 *  FILES ACCESSED     :  ACTIVEPROBLEMFILE, KRITERIAFILE  *
 *  EXTERNAL CALLS     :  GETTHEKEYS, ODOMETER             *
 *  EXTERNAL FILTERS   :  FILTER6.LIB, FILTER9.LIB         *
 *  CALLED FROM        :  WEEDHOPPER_MENU                  *
 *  PURPOSE            :  INITIALIZES VARIABLES, CHECKS    *
 *                        KRITERIAFILES                    *
 ***********************************************************)

var
   CHM : CHAR;
   TEMPALT : STRING[12];

   begin   {InitVariables}
      pt1 := 2;  pt2 := 2;  pt3 := 77;  pt4 := 21;
      window(pt1,pt2,pt3,pt4);               clrscr;
      Criteria.Flag1 := 0;  QuitFlg1 := 1;
      ShowMe := 0;    L := 0;
      criteria.flag2 := 0;  QuitFlg2 := 1;
      ThrNum := 1;    M := 0;
      criteria.flag3 := 0;  QuitFlg3 := 1;
      SecNum := 1;    N := 0;
      QuitFlag := 0;        Y := 1;            Count := 1;
      ChkFlag1 := 0;        Num := 1;          CHA := 'N';
      Startup := True;      CritLimit := 5;
      NewCritLimit := 10;
      StopGap := True;      Recount := 0;
      changerec := 'N';
      Scrollit := False;
      Assign(ActiveProblemFile,concat(FILEDRIVE,':Probs.txt'));
      Reset(ActiveProblemFile);
      repeat
         read(ActiveProblemFile,Members);
      until  (Members.Member = NameString) and
             (Members.Problem = ProbName) and
             (members.choice = alternative);
      NewString := Probname+alternative+'.'+Members.member;
      close(ActiveProblemFile);
      Assign(kriteriaFile,concat(FILEDRIVE,':',newstring));
```

```
reset(kriteriafile);
z := filesize(kriteriafile);
if z = 0 then
   begin
      Startup := False;    problemflag :=  'a';
      Criteria.Statflag := problemflag;
      members.CheckState := problemflag;
      members.checkchange := changerec;
      close(kriteriafile);
   end;
if z > 0 then
   begin
      reset(KriteriaFile);
      while not EOF(KriteriaFile) do
         begin   {While Statement}
            read(KriteriaFile,Criteria);
            problemflag := Criteria.StatFlag;
            odometer;
         end;    {While Statement}
      close(KriteriaFile);
   end;
case problemflag of
   'a'  :  if startup then
              begin   {If Statement}
                 if alternative = 'A' then
                    tempalt := 'Alternatives
                 else
                    tempalt :=  'Criteria';
                 gotoXY(21,11);
                 Write('Do you wish to review your
                        ',tempalt,'?  ');
                 gotoxy(65,11);
                 repeat
                    getthekeys(Inputstring,1);
                    cha := inputstring;
                    gotoxy(61,11);
                    chm := cha;
                 until chm in ['Y', 'N'];
                 clrscr;
              end;    {If Statement}
   'b'  :  begin   {If Statement}
              gotoXY(15,6);
              Write('You are entering the Sub Criteria
                     level.  If ');
              gotoXY(15,7);
              Write(' this is the initial entry, you
                     may review the');
              gotoXY(15,8);
              Write('last level of criteria, but you
                     may not change');
                    gotoXY(15,9);
                    Write('it.  However you may review the
                              criteria you');
                    gotoXY(15,10);
                    Write('have already entered at this
                              level and change');
                    gotoXY(15,11);
                    Write('that.  Do you wish to review your
                              criteria?  ');
                    gotoxy(61,11);
                    repeat
                       getthekeys(Inputstring,1);
                       cha := inputstring;
                       gotoxy(61,11);
                       chm := cha;
                    until chm in ['Y','N'];
                    clrscr;
                 end;     {If Statement}
         'c'  :  begin    {If Statement}
                    gotoXY(14,6);
                    Write('You are entering the Tertiary ,
                    ' Criteria level.  If ');
                    gotoXY(14,7);
                    Write('this is the initial entry, you
                              may review the');
                    gotoXY(14,8);
                    Write('last level of criteria, but you
                              may not change');
                    gotoXY(14,9);
                    Write('it.  However you may review the
                              criteria you');
                    gotoXY(14,10);
                    Write('have already entered at this
                              level and change');
                    gotoXY(14,11);
                    Write('that.  Do you wish to review your
                              criteria?  ');
                    gotoxy(61,11);
                    repeat
                       getthekeys(Inputstring,1);
                       cha := inputstring;
                       gotoxy(61,11);
                       chm := cha;
                    until chm in ['Y','N'];
                    clrscr;
                 end;     {If Statement}
        'h','k','n','q','j','m','p' :
           begin    {Inside case Statement}
              gotoXY(15,7);
              Write('Your flag has been set stating that you
                        have ');
              gotoXY(15,8);
              Write('finished inputing criteria at the last
                        level.');
              gotoXY(15,9);
```

```
          Write('You may not enter any more criteria at
                 this');
          gotoXY(15,10);
          Write('time.  However you may review the
                 criteria you');
          gotoXY(15,11);
          Write('have already entered, but you may not
                 change it.');
          gotoXY(15,12);
          Write('Press Return to continue.');
          cha := 'Y';
          getthekeys(Inputstring,1);
          clrscr;
       end;    {Inside case Statement}
    'i','l','o' :
       begin   {Inside case Statement}
          gotoXY(15,7);
          Write('All members of the committee have
                 finished ');
          gotoXY(15,8);
          Write('entering their criteria.  You may now
                 review');
          gotoXY(15,9);
          Write('all criteria that has been entered.  Be
                 advised');
          gotoXY(15,10);
          Write('that this procedure will be repeated
                 until there');
          gotoXY(15,11);
          Write('is a resolution between all members
                 concerning');
          gotoXY(15,12);
          Write('what criteria is to be kept.  Press
                 RETURN to');
          gotoXY(15,13);
          Write('continue.');
          cha := 'Y';
          getthekeys(Inputstring,1);
          clrscr;
       end;    {Inside case Statement}
    end;   {case statement}
 end;    {InitVariables}
```

```
procedure Ritebox;

(************************************************************
 *  PROCEDURE          :  RITEBOX                            *
 *  SUPPORTS PROGRAM   :  CTOUCH.PAS                         *
 *  LOCAL VARIABLES    :  NONE                               *
 *  GLOBAL VARIABLES   :  CHATOK, ALTERNATIVE.               *
 *  ARRAYS USED        :  NONE                               *
 *  FILES ACCESSED     :  NONE                               *
 *  EXTERNAL CALLS     :  BASICBOX                           *
 *  EXTERNAL FILTERS   :  FILTER1.LIB                        *
 *  CALLED FROM        :  WEEDHOPPER_MENU                    *
 *  PURPOSE            :  SETS UP THE INITIAL ODOMETER       *
 *                        SCREEN AND WRITES                  *
 *                        PRELIMINARY DATA TO SCREEN.        *
 ************************************************************)

   begin {Ritebox}
      clrscr;                 ChatOK := False;
      window(1,1,78,25);
      Clrscr;                 basicbox(1,1,78,22);
      port[$03d9]:= $f and 1;
      GotoXY(2,23);        clreol;       GotoXY(28,23);
      clreol;
      if alternative = 'A  then
         begin
           GotoXY(28,23);  Write('Alternative Development  ;
           GotoXY(17,24);
           Write   ('Input   Final   Holding   Review
                     Alternatives');
         end
      else
        begin
           GotoXY(28,23);  Write('Criteria Level of Entry ');
           GotoXY(2,24);
           Write(' Major  Sub Criteria  Tertiary Criteria
                   Final  Holding .
                  ' Review Criteria');
        end;
      gotoXY(2,25);    write('F1=Help  F2=CHATTERBOX
                                F3=Problem  ');
      gotoXY(37,25);  write('Explanation  F10=Quit');
      gotoxy(30,1);  textbackground(red);
      textcolor(yellow);
      write('      TOUCHSTONE        ';
      textbackground(blue);
      textcolor(white);
   end;    {Ritebox}
```

```
procedure MainCriteria;

(**********************************************************
 *  PROCEDURE          :  MAINCRITERIA                     *
 *  SUPPORTS PROGRAM   :  CTOUCH.PAS                       *
 *  LOCAL VARIABLES    :  SHORTNAME, LONGNAME              *
 *  GLOBAL VARIABLES   :  PT1, PT2, PT3, PT4, PROBLEMFLAG, *
 *                        QUITFLAG,                        *
 *                        CRITERIA, QUITFLG1, NUM, SECNUM, *
 *                        QUITFLG2,                        *
 *                        THRNUM, QUITFLG3, INPUTSTRING,   *
 *                        MOVEX, STOPPROG,                 *
 *                        CRITLIMIT, NEWCRITLIMIT, COUNTED, *
 *                        L, M, N, A                       *
 *  ARRAYS USED        :  NONE                             *
 *  FILES ACCESSED     :  KRITERIAFILE                     *
 *  EXTERNAL CALLS     :  GETTHEKEYS                       *
 *  EXTERNAL FILTERS   :  FILTER9.LIB                      *
 *  CALLED FROM        :  MAINCRITERIA                     *
 *  PURPOSE            :  ALLOWS THE COMMITTE MEMBER TO ADD *
 *                        ALTERNATIVES/CRITERIA TO A NEW OR *
 *                        EXISTING FILE.                   *
 **********************************************************)

var
   SHORTNAME   :  STRING[10];
   LONGNAME    :  STRING[58];

   begin    {MainCriteria}
      pt1 := 2;  pt2 := 2;  pt3 := 77;  pt4 := 21;
      window(pt1,pt2,pt3,pt4);
      if problemflag <> 'a' then
         begin  {If statement}
            seek(kriteriafile,recount-1);
            read(kriteriafile,criteria);
         end;   {If statement}
      repeat
         if (QuitFlag = 0) and (problemflag < 'e') then
            begin  {If statement within Repeat}
      case ProblemFlag of
         'a'  :  begin   {A statement within Case}
                    if criteria.flag1 = 0 then
                       GotoXY(1,1)
                    else
                       GotoXY(1,whereY);
                    Write(Num,'.    ');      Num := Num + 1;
                    movex := wherex;
                    QuitFlg1 := QuitFlg1 + 1;
                    Criteria.Flag1 := Criteria.Flag1 + 1;
                 end;    {A statement within Case}
         'b'  :  begin   {B statement within Case}
                    GotoXY(4,wherey);  Write(SecNum, '.    ');
                    movex := wherex;
                    SecNum := Succ(SecNum);
                    QuitFlg2 := QuitFlg2 + 1;
                    Criteria.Flag2 := Criteria.Flag2 + 1;
                 end;     {B statement within Case}
      'c'  :  begin    {C statement within Case}
                    GotoXY(8,wherey);  Write(ThrNum,').   '):
                    movex := wherex;
                    ThrNum := ThrNum + 1;
                    QuitFlg3 := QuitFlg3 + 1;
                    Criteria.Flag3 := Criteria.Flag3 + 1;
                 end;     {C statement within Case}
   end;  {Case Statement}
   repeat
      getthekeys(Inputstring,10);
      shortName := inputstring;
      gotoxy(movex,wherey);
   until (ord(shortname[1]) > 32) or (stopprog);
   a := 2;
   criteria.critname := shortName[1];
   while (shortname[a] <> chr(13)) and (a<11) do
      begin
         criteria.critname :=  concat(criteria.critname,
                                 shortname[a]);
         a := a + 1;
      end;
   writeln;
   if not StopProg and not (QuitFlg2 > CritLimit + 1) and
      not (QuitFlg3 > CritLimit + 1) and
      not (QuitFlg1 > NewCritLimit + 1) then
      begin  {Load file}
         GotoXY(2,wherey);            Write('Define:   ');
         movex := wherex;
         repeat
            getthekeys(Inputstring,58):
            longName := inputstring;
            gotoxy(movex,wherey);
         until (ord(longname[1]) > 32) or (stopprog):
         a := 2;
         criteria.critdef := longName[1];
         while (longname[a] <> chr(13)) and
               (a<counted+1) do
            begin
               criteria.critdef :=
                  concat(criteria.critdef,longname[a]);
               a := a + 1;
            end;
         writeln:
         l := Criteria.Flag1 * 100;
         m := Criteria.Flag2 * 10;
         n := Criteria.Flag3;
         Criteria.CheckPoint := l + m + n;
         seek(kriteriafile,filesize(kriteriafile));
         Write(Kriteriafile.Criteria);
      end;  {Load file}
      end;  {If statement within repeat}
      until StopProg or (QuitFlg1 > NewCritLimit) or
            (QuitFlg2 > CritLimit) or
            (QuitFlg3 > CritLimit);
   end;      {MainCriteria}


procedure Window3;

(***********************************************************
 *  PROCEDURE           :  WINDOW3                          *
 *  SUPPORTS PROGRAM    :  CTOUCH.PAS                       *
 *  LOCAL VARIABLES     :  CHM                              *
 *  GLOBAL VARIABLES    :  PROBLEMFLAG, RECOUNT, Z, CRITERIA,*
 *                         NUM, SECNUM,                     *
 *                         THRNUM, QUITFLG1, QUITFLG2,      *
 *                         QUITFLG3, STOPGAP,               *
 *                         CHKFLAG1, CHKFLAG2, CHKFLAG3,    *
 *                         SHOWME, QUITFLAG,                *
 *                         INPUTSTRING, FLAGCHOICE, NAMES,  *
 *                         LIMMIT                           *
 *  ARRAYS USED         :  NONE                             *
 *  FILES ACCESSED      :  KRITERIAFILE                     *
 *  EXTERNAL CALLS      :  ODOMETER, FINALCHOICE, LOADARRAY, *
 *                         NEWWRITE, CHATRCHECK,            *
 *                         RANTOCOMPLETION, MAINCRITERIA,   *
 *                         GETTHEKEYS                       *
 *  EXTERNAL FILTERS    :  FILTER6.LIB, FILTER9.LIB         *
 *  CALLED FROM         :  WEEDHOPPER_MENU                  *
 *  PURPOSE             :  LISTS ALTERNATIVES/CRITERIA WHEN *
 *                         THE USER HAS PREVIOUSLY INPUT    *
 *                         DATA BUT DOES NOT WANT TO REVIEW *
 *                         THAT DATA.                       *
 ***********************************************************)

var
   CHM : CHAR;

   begin     {Window3}
      clrscr;       Odometer;       chatrcheck;
      recount := 0;      reset(kriteriafile);
      z := filesize(kriteriafile);
      if (problemflag > 'a') and (problemflag < 'e') then
         begin    {If Statement}
            repeat    {Main Repeat Module}
               seek(kriteriafile,recount);
               read(Kriteriafile,criteria);
               repeat    {Embedded Repeat Module}

      (*****************************************************
      * Writing Major Criteria, (X000), previously entered *
      * when problemflag = a.  ProblemFlag = b for this    *
      * module to be activated, and allows subcriteria to  *
      *  be entered, (XX00), X's being integers.           *
      *****************************************************)
   case Criteria.flag1 of
      1..100 :  begin   {inside case statement flag1}
                if (Criteria.flag2 = 0) and
                   (Criteria.Flag3 = 0) then
                begin  {Case If Statement}
                   if criteria.flag1 = 1 then
                      GotoXY(1,1)
                   else
                      GotoXY(1,whereY);
                   Write(Num,'.   ');
                   ThrNum := 1;   Secnum := 1;
                   Num := Num + 1;
                   QuitFlg2 := 1;
                   QuitFlg1 := QuitFlg1 + 1;
                end;   {Case If Statement}

(*****************************************************
* Writing Sub Criteria, (XX00), previously entered   *
* when problemflag = b.  ProblemFlag = c for this    *
* module to be activated, and allows tertiary        *
* criteria to be entered, (XXX0), X's being integers.*
*****************************************************)

  case Criteria.flag2 of
     1..100 : begin   {inside case statement flag2}
              if (Criteria.flag3 = 0) then
              begin  {Case If Statement}
                 gotoXY(4,wherey);
                 Write(SecNum,'.   ');
                 SecNum := Succ(SecNum);
                 QuitFlg2 := QuitFlg2 + 1;
                 ThrNum := 1;
                 if QuitFlg2 = CritLimit then
                    StopGap := False;
                 QuitFlg3 := 1;
              end;   {Case If Statement}

(*****************************************************
 * Writing Tertiary Criteria, (XXX0), previously     *
 * entered when problemflag = c.  ProblemFlag = d    *
 * for this module to be activated, and allows       *
 * tertiary criteria to be entered, (XXXX), X's      *
 * being integers.                                   *
 ****************************************************)

  case Criteria.flag3 of
     1..100  :  begin  {Case If Statement}
                   gotoXY(8,wherey);
                   Write(ThrNum,').   ');
                   ThrNum := ThrNum + 1;
                   QuitFlg3 := QuitFlg3 + 1;
                   if QuitFlg3 = CritLimit then
                      StopGap := False;
              end;   {Case If Statement}
  end;  {Case Statement flag3}
```

```
                 end;    {inside case statement flag2}
         end;  {Case Statement flag2}
                   writeln(Criteria.CritName,':
                        ',Criteria.CritDef);
                 end;    {inside case statement flag1}
         end;  {Case Statement flag1}
                   if (ProblemFlag = 'c') and
                      (Criteria.Flag1 = ChkFlag1) and
                      (Criteria.Flag2 > ChkFlag2) and
                      (Criteria.Flag3 = 0) and
                      (ChkFlag3 = 0) then
                        Showme := 1;
               ChkFlag1 := Criteria.Flag1;
               ChkFlag2 := Criteria.Flag2;
               ChkFlag3 := Criteria.Flag3;
               recount := recount + 1;
               if recount < z then
                  read(Kriteriafile,criteria);
               if (ProblemFlag = 'c') then
                  begin   {C If Statement}
                    if (Criteria.Flag2 > ChkFlag2) and
                       (Criteria.Flag3 = 0) and
                       (ChkFlag2 > 0) and
                       (ChkFlag3 = 0) then
                       Showme := 1;
                    if (Criteria.Flag2 > ChkFlag2) and
                       (Criteria.Flag3 = 0) and
                       (ChkFlag3 > 0) then
                       Showme := 1;
                    if (Criteria.Flag2 = ChkFlag2) and
                       (Criteria.Flag3 > 0) and
                       (ChkFlag3 = 0) then
                       Showme := 0;
                    if (Criteria.Flag2 = ChkFlag2) and
                       (Criteria.Flag3 = 0) and
                       (ChkFlag3 > 0) then
                       Showme := 1;
                  end;   {C If Statement}
             until (Criteria.Flag1 > Chkflag1) or
                   (Showme = 1) or
                   (recount = z);
             if (QuitFlg2 > CritLimit) or
                (QuitFlg3 > CritLimit) then
                QuitFlag := 1;
             MainCriteria;
             Showme := 0;    QuitFlg1 := 1;
             QuitFlg2 := 1;
             QuitFlg3 := 1;  QuitFlag := 0;
          until (recount = z);
       end     {If Statement}
    else
       if problemflag < 'e' then
       begin   {If/Else Statement}
          while not EOF(kriteriafile) do
             begin   {While Statement}
```

```
                  read(Kriteriafile,criteria);
                  case Criteria.flag1 of
                     1..100 :  begin   {inside case
                                         statement flag1}
                               if (Criteria.flag2 = 0) and
                                  (Criteria.Flag3 = 0) then
                               begin   {Case If Statement}
                                 if criteria.flag1 = 1 then
                                    GotoXY(1,1)
                                 else
                                    GotoXY(1,whereY);
                                  Write(Num,'.  ');
                                  Num := Num + 1;
                                  QuitFlg1 := QuitFlg1 + 1;
                                  Secnum := 1;
                               end;    {Case If Statement}
                  case Criteria.flag2 of
                     1..100  :  begin    {inside case
                                           statement flag2}
                              if (Criteria.flag3 = 0) then
                              begin  {Case If Statement}
                                 gotoXY(4,wherey);
                                 Write(SecNum,'.  ');
                                 SecNum := Succ(SecNum);
                                 QuitFlg2 := QuitFlg2 + 1;
                                 ThrNum := 1;
                              end;   {Case If Statement}
                  case Criteria.flag3 of
                     1..100  :  begin  {Case If Statement}
                                 gotoXY(8,wherey);
                                 Write(ThrNum,').  ');
                                 ThrNum := ThrNum + 1;
                                 QuitFlg3 := QuitFlg3 + 1;
                              end;   {Case If Statement}
                  end;  {Case Statement flag3}
                              end;    {inside case statement
                                        flag2}
                  end;  {Case Statement flag2}
                        Writeln(Criteria.CritName,':
                               ',Criteria.CritDef);
                               end;    {inside case
                                         statement flag1}
                  end;  {Case Statement flag1}
                  if QuitFlg1 = NewCritLimit then
                     StopGap := False;
                  ChkFlag1 := Criteria.Flag1;
               end;    {While Statement}
            if not (QuitFlg1 > NewCritLimit) then
               Maincriteria;
         end;    {If/Else Statement}
     close(kriteriafile);
```

```
      if problemflag <> 'z' then
         begin
            gotoXY(1,19);
            write('Are you finished with this level of
                   criteria, ',
                   'or will you be entering more?');
            gotoXY(1,20);
            write('Enter ''F'' for Finished, or ''M'' for
                   More:  ');
            gotoxy(45,20);
            repeat
               getthekeys(Inputstring,1);
               flagchoice := inputstring;
               chm := flagchoice;
               gotoxy(45,20);
            until chm in ['F','M'];
            if (FlagChoice = 'F') then
                FinalChoice;
         end;
         if problemflag = 'z' then
            rantocompletion;
      LoadArray;      NewWrite(Names,Limmit);
      chatrcheck;
   end;    {Window3}


procedure WEEDHOPPER_MENU;

(**********************************************************
 *  PROCEDURE           :  WEEDHOPPER_MENU                 *
 *  SUPPORTS PROGRAM    :  CTOUCH.PAS                      *
 *  LOCAL VARIABLES     :  CH, SELECTION, CONTINUE,        *
 *                         COMPLETED, FILECHECK,           *
 *                         SHORTNAME, TEMPDEFINITION, COUNTS *
 *  GLOBAL VARIABLES    :  COMPLETED, WEEDDEF, FILECHECK, Y, *
 *                         X, MARKER,                      *
 *                         MOVEOVER, FILEDRIVE, Z, LINEMARK, *
 *                         MEMBERS, NAMESTRING, ALTERNATIVE, *
 *                         INPUTSTRING, PROBNAME, DATE,    *
 *                         COUNT, NAMES, LIMMIT, CHATOK    *
 *  ARRAYS USED         :  NONE                            *
 *  FILES ACCESSED      :  ACTIVEPROBLEMFILE, DATEFILE     *
 *  EXTERNAL CALLS      :  INTROSCREEN, GETTHEKEYS, RITEBOX, *
 *                         CHATRCHECK,  INITVARIABLES,     *
 *                         LOADARRAY, REVIEW, WINDOW3,     *
 *                         LOADEMUP                        *
 *  EXTERNAL FILTERS    :  FILTER2.LIB FILTER7.LIB,        *
 *                         FILTER9.LIB, TAILEND.LIB        *
 *  CALLED FROM         :  MAIN BODY OF PROGRAM CTOUCH.PAS *
 *  PURPOSE             :  GIVES THE COMMITTEE MEMBER THE  *
 *                         OPPORTUNITY TO EITHER REVIEW PAST *
 *                         ENTRIES OR START NEW ONES.      *
 **********************************************************)
```

```
var
  CH, SELECTION                           :  CHAR;
  CONTINUE, COMPLETED, FILECHECK          :  BOOLEAN;
  DATEFILE                                :  TEXT;
  SHORTNAME                               :  STRING[7];
  TEMPDEFINITION                          :  STRING[58];
  COUNTS                                  :  INTEGER;

   begin   {procedure WeedHopper_MENU}
      COMPLETED := false;
      repeat
         weeddef := false;
         FILECHECK := False;
         INTROSCREEN;
         gotoxy(18,1);   write ('COMMITTEE MEMBER MENU');
         gotoxy(1,3);
         write ('At the present time, you are a member on
                 committees ');
         gotoxy(1,4);
          write ('discussing the following problems:');
         Y := 6;      X := 1;      Marker := 0;
         MoveOver := 13;
         Assign(ActiveProblemFile,concat(FILEDRIVE,':Probs.txt'));
         {$I-}
         Reset(ActiveProblemFile);
         {$I+}
         z := 0;
         if IOresult = 0 then
            begin   {I/O result}
               while not EOF(ActiveProblemfile) do
                  begin   {While not EOF Loop}
                     LineMark := False;
                     Read(ActiveProblemFile, Members);
                     if (members.member = namestring) and
                        (members.choice = alternative) then
                        begin
                           LineMark := True;
                           gotoXY(X,Y);
                           Write(Members.Problem);
                           z := succ(z);
                        end;
                     if Y > 13  then
                        begin   {if Y > 13}
                           case marker of
                              1  :  moveover := 25;
                              2  :  moveover := 37;
                              3  :  moveover := 49;
                           end;
                           X := MoveOver;  Y := 6;
                           Marker := Marker + 1;
                        end      {If Y > 13}
                     else
                        if LineMark then
                           Y := Y + 1;
                  end;    {While not EOF Loop}
```

```
close(ActiveProblemfile);
if z = 0  then
   begin
      gotoxy(8,8);
      write('You are not currently serving on a
             committee');
      delay(4000);
      completed := true;
   end;
if not completed then
   begin
      gotoxy(1,15);
      write ('SELECTION CHOICES:  1)  Choose a
               problem');
      gotoxy(46,15);
      write ('2)  Exit to DOS');
      gotoxy (22,16);
      write ('SELECTION:  *');    gotoxy(34,16);
      repeat
         read(kbd,CH);
      until CH in ['1','2'];
      write(CH);
      delay(1000);
   end;
if CH = '1' then
   begin {select choice}
      counts := 0;
      repeat
         GotoXY(22,16);
         Write('Choose the problem:  ');
         repeat
            getthekeys(Inputstring,7);
            shortName := inputstring;
            gotoxy(43,16);
         until shortname[1] > #32;
           {remove spaces from shortname}
         a := 2;
         probname := shortName[1];
         while (shortname[a] <> chr(32)) and
               (a<8) do
            begin
               probname :=
               concat(probname,shortname[a]);
               a := a + 1;
            end;  {while shortname[a]}
    {gets the date from a file DATE.TXT}
         assign(datefile,'date.txt');
         reset(datefile);
         readln(datefile,date);
         close(datefile);
         reset(activeproblemfile);
         count := 1;
         while not EOF(activeproblemfile) do
            begin   {while statement}
               Read(ActiveProblemFile,Members);
         if (Members.Problem = ProbName)
         and (Members.Member = NameString)
         and (members.choice = alternative)
         then
             begin
                filecheck := true;
                members.dateline := date;
                tempdefinition :=
                members.definition;
             end;
         seek(activeproblemfile,count-1);
         write(activeproblemfile,members);
         count := succ(count);
      end;    {while statement}
   close(ActiveProblemfile);
   counts := succ(counts);
until (filecheck) or (counts > 2);
if not (filecheck) then
   begin
      clrscr;  gotoXY(9,8);
      Writeln('I''m sorry but you don''t
               seem to be typing');
      gotoXY (9,9);
      Write('in a problem that we have on
            file.');
      delay(4000);
      completed := true;
   end     {If Statement}
else
   begin
   clrscr;
   gotoxy(10,3);
   WeedDef := true;
   write('A short, one line definiton of
         ',probname,
         ' follows.');
   gotoXY (2,6);  writeln(TempDefinition);
   gotoxy(10,8);
   write('If at any time you wish to see a
          more in ');
   gotoxy(10,10);
   write('depth explanation of the
          problem, press F3');
   gotoxy(15,16);
   write('Press Return to continue '):
   getthekeys(Inputstring,1);
   RITEBOX;
   CHATOK := true;
   CHATRCHECK;
   INITVARIABLES;
      if (CHA = #89) and (STARTUP) then
         begin
            LOADARRAY;
            REVIEW (NAMES,LIMMIT);
         end   {if CHA=#89}
```

```
                              else
                                 WINDOW3;
                           end;    {If/Else}
                     end    {if CH=1}
                  else
                     COMPLETED := true;
                  end  {if IOresult = 0}
               else
                  begin
                     clrscr;                       gotoxy (13,8);
                     write ('File PROBS.TXT not found on drive
                           ',FILEDRIVE);
                     sound(800);                   delay(500);nosound;
                     close(ActiveProblemfile);
                     delay(2000);                  COMPLETED := true;
                  end;  {else}
            until COMPLETED;
            loademup;
         end;  {procedure WeedHopper_MENU}


begin     {Main Program}
  INVOCATOR := 'W';
  GETFILENAMES;
  assign (TEMPFILE,'DRIVEFIL.TMP');
  {$I-}
  reset (TEMPFILE);
  {$I+}
  if IOresult = 0 then
    erase (TEMPFILE);
  if (AUTHORIZED) and (Invocator = 'W') then begin
    INTROSCREEN;
    gotoxy(10,8);
    write('ACCESS APPROVED - WELCOME TO TOUCHSTONE!');
    delay(3000);
    ALTERNATECHOICE;
     (******* call touchstone programs *******)
         if INVOCATOR = 'W' then
           WEEDHOPPER_MENU;
     (****************************************)
    INTROSCREEN;
    gotoxy (4,8);
    write ('THANK YOU FOR USING TOUCHSTONE - HAVE A NICE
            DAY!');
    delay (2000);
  end;   {if AUTHORIZED}
  loadthefiles;
end.      {Main Program}
```

```
program FLAGSET(INPUT,OUTPUT);

(************************************************************
 *  PROGRAM            :  FLAGSET.PAS                       *
 *  ARRAYS USED        :  CRITARRAY                         *
 *  FILES ACCESSED     :  TEMPFLAGSET, ACTIVEPROBLEMFILE,   *
 *                        KRITERIAFILE                      *
 *  EXTERNAL CALLS     :  BUBBLESORT, CRITSORT              *
 *  EXTERNAL FILTERS   :  FILTER6.LIB                       *
 *  CALLED FROM        :  TS.BAT                            *
 *  PURPOSE            :  MERGES THE ALTERNATIVES/CRITERIA  *
 *                        OF ALL MEMBERS WHEN CERTAIN       *
 *                        VARIABLES ARE MATCHED.  USED AS A *
 *                        COM FILE AT THE LAST OF THE BATCH *
 *                        FILE TS.BAT.  NO INTERACTION FROM *
 *                        THE USER IS REQUIRED.  THE  LAST  *
 *                        ACT OF THIS PROGRAM IS TO SET THE *
 *                        SCREEN BACK TO NORMAL.            *
 ************************************************************)

type
   STRING3               = STRING[3];
   STRING8               = STRING[8];
   STRING10              = STRING[10];
   STRING12              = STRING[12];

   PROBREC               = record
                              CHECKSTATE        : CHAR;
                              CHECKCHANGE       : CHAR;
                              CHOICE            : CHAR;
                              PROBLEM           : STRING[7];
                              NUMMEMS           : INTEGER;
                              MEMBER            : STRING3;
                              DEFINITION        : STRING[58];
                              DATELINE          : STRING12;
                           end;


   CRIREC                = record
                              FLAG1          : INTEGER;
                              FLAG2          : INTEGER;
                              FLAG3          : INTEGER;
                              CHECKPOINT     : INTEGER;
                              STATFLAG       : CHAR;
                              CRITNAME       : STRING10;
                              CRITDEF        : STRING[58];
                           end;

   CRITARRAY                 = array[1..150] of CRIREC;
   PROBARRAY                 = array[1..200] of PROBREC;
```

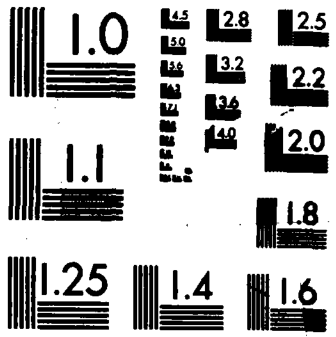
```
var

   FLAGGED, FILEDRIVE,
   PROBLEMFLAG, CHANGEREC                  :  CHAR;

   STARTMERGE, ONCECOUNTED,
   ANONYMOUS, CHANGEFLAG                   :  BOOLEAN;

   PT1, L, COUNT,  FLAGCOUNT, I            :  INTEGER;
   PT2, M, LIMID,  KEEPTOGETHER            :  INTEGER;
   PT3, N, TRACK1, DOUBLECOUNTED           :  INTEGER;
   PT4, Z, LIMMIT, COUNTED, FLAGEND        :  INTEGER;

   ALTERNATIVE                             :  STRING[1];
   NAMESTRING                              :  STRING3;
   PROBNAME                                :  STRING[7];
   NEWSTRING, DATE                         :  STRING12;

   NAMES                                   :  CRITARRAY;
   PROBS                                   :  PROBARRAY;

   CRITERIA                                :  CRIREC;
   MEMBERS                                 :  PROBREC;

   KRITERIAFILE                            :  file of CRIREC;
   ACTIVEPROBLEMFILE                       :  file of PROBREC;

{$IFILTER6.LIB}


procedure PutEmTogether;

(**********************************************************
 *  PROCEDURE          :  PUTEMTOGETHER                    *
 *  SUPPORTS PROGRAM   :  FLAGSET.PAS                      *
 *  LOCAL VARIABLES    :  NONE                             *
 *  GLOBAL VARIABLES   :  NEWSTRING, PROBNAME, ALTERNATIVE, *
 *                        MEMBERS, FILEDRIVE, COUNTED,     *
 *                        LIMMIT, KEEPTOGETHER             *
 *  ARRAYS USED        :  CRITARRAY                        *
 *  FILES ACCESSED     :  KRITERIAFILE                     *
 *  EXTERNAL CALLS     :  NONE                             *
 *  EXTERNAL FILTERS   :  NONE                             *
 *  CALLED FROM        :  FLAGSETTER                       *
 *  PURPOSE            :  LOADS AN ARRAY WITH ALL FILES    *
 *                        HAVE THE SAME PROBLEM NAME.      *
 **********************************************************)

   begin    {PutEmTogether}

      NewString := Probname+alternative+'.'+members.member;

      Assign(kriteriaFile,concat(FILEDRIVE,':',newstring));
      reset(Kriteriafile);
      while not EOF(KriteriaFile) do

         begin   {While Statement}

            Read(KriteriaFile,Names[Counted]);

            names[counted].checkpoint :=
            names[counted].checkpoint + KeepTogether;

            Counted := Counted + 1;

         end;    {While Statement}

      Limmit := Counted;

      close(KriteriaFile);

      KeepTogether := KeepTogether + 1;

   end;     {putemtogether}


procedure AllTogether(var Names : CritArray;  Limmit :
integer);

(************************************************************
 *  PROCEDURE         :  ALLTOGETHER                        *
 *  SUPPORTS PROGRAM  :  FLAGSET.PAS                        *
 *  LOCAL VARIABLES   :  NONE                               *
 *  GLOBAL VARIABLES  :  COUNTED, NAMES, LIMMIT,            *
 *                       DOUBLECOUNTED, PROBLEMFLAG,        *
 *                       NEWSTRING, FILEDRIVE, PROBNAME,    *
 *                       ALTERNATIVE, ONCECOUNTED           *
 *  ARRAYS USED       :  CRITARRAY                          *
 *  FILES ACCESSED    :  KRITERIAFILE                       *
 *  EXTERNAL CALLS    :  BUBBLESORT, CRITSORT               *
 *  EXTERNAL FILTERS  :  FILTER6.LIB                        *
 *  CALLED FROM       :  FLAGSETTER                         *
 *  PURPOSE           :  THIS PROCEDURE RELOADS EACH        *
 *                       USER'S FILE WITH ALL OF THE        *
 *                       CRITERIA THAT EACH USER INITIALLY  *
 *                       ENTERED.  EACH USER THEN WILL      *
 *                       HAVE THE SAME IDENTICAL FILE TO    *
 *                       DELETE, CHANGE OR ADD TO IN        *
 *                       SELECTING THE FINAL CRITERIA.      *
 ***********************************************************)

   begin   {alltogether}

      if OnceCounted then

         begin

            bubblesort(names,Limmit);
```

```
          counted := 1;                 doublecounted := 1;

          repeat

             if (Names[counted].Critname =
                 Names[counted+1].critname) and
                (names[counted].critdef =
                 names[counted+1].critdef) then
                 begin
                    Names[counted].Flag1 := 0;
                    doublecounted := doublecounted + 1;
                 end;

             counted := counted + 1;

          until counted = Limmit;

          Critsort(names,Limmit);

       end;

    counted := 1;

    NewString := Probname+alternative+'.'+members.member;

    Assign(kriteriaFile,concat(FILEDRIVE,':',newstring));
    rewrite(Kriteriafile);

    repeat

             names[counted].statflag := problemflag;

       if names[counted].flag1 > 0 then
          begin
             Write(kriteriafile,Names[Counted]);
          end;

       counted := counted + 1;

    until counted = Limmit;

    close(kriteriafile);

    OnceCounted := False;

 end;     {alltogether}
```

```
procedure FlagSetter;

(*************************************************************
 *  PROCEDURE          :  FLAGSETTER                         *
 *  SUPPORTS PROGRAM   :  FLAGSET.PAS                        *
 *  LOCAL VARIABLES    :  PROBLEMHOLD                        *
 *  GLOBAL VARIABLES   :  FILEDRIVE, ALTERNATIVE,            *
 *                        NAMESTRING, PROBNAME, COUNTED,     *
 *                        FLAGEND, KEEPTOGETHER, STARTMERGE, *
 *                        COUNT, CHANGEFLAG, FLAGCOUNT,      *
 *                        ONCECOUNTED, MEMBERS, FLAGGED,     *
 *                        PROBLEMFLAG                        *
 *  ARRAYS USED        :  NONE                               *
 *  FILES ACCESSED     :  TEMPFLAGSET, ACTIVEPROBLEMFILE     *
 *  EXTERNAL CALLS     :  PUTEMTOGETHER, ALLTOGETHER         *
 *  EXTERNAL FILTERS   :  NONE                               *
 *  CALLED FROM        :  MAIN BODY OF PROGRAM FLAGSET.PAS   *
 *  PURPOSE            :  THIS PROCEDURE SETS THE FLAGS SO   *
 *                        THAT THE USER CAN TELL WHERE HE    *
 *                        IS PERSONALLY AT IN THE SELECTION  *
 *                        PROCESS.                           *
 *************************************************************)

var
   PROBLEMHOLD             :  CHAR;
   TEMPFLAGSET             :  TEXT;

   begin   {FlagSetter}

      assign(tempflagset,'flagset.txt');
      reset(tempflagset);

      readln(tempflagset,filedrive);
      readln(tempflagset,alternative);
      readln(tempflagset,namestring);
      readln(tempflagset,probname);
      close(tempflagset);

      erase(tempflagset);

      Assign(ActiveProblemFile,concat(FILEDRIVE,':Probs.[illegible]
      reset(ActiveProblemFile);

      counted := 1;          count := 0;
      flagcount := 1;
      flagend := 1;          Startmerge := False;
      OnceCounted := True;
      KeepTogether := 1;     ChangeFlag := True;

      while not EOF(activeproblemfile) do

         begin   {while statement

            read(ActiveProblemFile,[illegible]
```

AD-A183 203 TOUCHSTONE: A CRITERIA DEVELOPMENT PROGRAM FOR GROUP DECISION SUPPORT SYSTEMS(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA R T WOOLDRIDGE ET AL. MAR 87 2/3

UNCLASSIFIED F/G 12/5 NL


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6

```
          if (members.Problem = ProbName) and
             (members.choice = alternative) then

             begin

                if members.member = namestring then
                   Flagged := members.checkstate;

                flagend := flagend + 1;

             end;

          count := count + 1;

       end;    {while statement}

    close(ActiveProblemFile);

    flagcount := 1;

   {Check to see if the members are at the same stage}

    reset(activeproblemfile);

    while not EOF(ActiveProblemFile) do

       begin    {While Statement}

          read(ActiveProblemFile,members);

          if (members.checkchange = 'C') and
             (members.problem = probname) and
             (members.choice = alternative) then
             changeflag := false;

          if (members.Checkstate = Flagged) and
             (members.problem = probname) and
             (members.choice = alternative) then
              Flagcount := Flagcount + 1;

       end;  {while statement}

    close(ActiveProblemFile);

    reset(activeproblemfile);

    if FlagCount = flagend then

       begin    {If Statement}

          while not EOF(ActiveProblemFile) do

             begin    {While Statement}

                read(ActiveProblemFile,members);
```

```
if (members.Problem = ProbName) and
   (members.choice = alternative) then

   begin   {Embedded If Statement}

      case members.CheckState of

         'h'  :  begin
                    PutEmTogether;
                    Startmerge := True;
                    members.CheckState
                          := 'i';
                    problemflag := 'i';
                 end;

         'j'  :  begin
                    PutEmTogether;
                    Startmerge := True;
                    if changeflag then
                       begin
                        if members.choice
                           = 'A' then
                             begin
                       members.CheckState
                           := 'z';
                       problemflag := 'z';
                             end
                          else
                             begin
                       members.CheckState
                           := 'b';
                       problemflag := 'b';
                             end;
                       end
                    else
                       begin
                       members.CheckState
                           := 'i';
                       problemflag := 'i';
                       end;
                 end;

         'k'  :  begin
                    PutEmTogether;
                    Startmerge := True;
                    members.CheckState
                          := 'l';
                    problemflag := 'l';
                 end;

         'm'  :  begin
                    PutEmTogether;
                    Startmerge := True;
                    if changeflag then
```

```
                                                          begin
                                                          members.CheckState
                                                                 := 'c';
                                                          problemflag := 'c';
                                                          end
                                                      else
                                                          begin
                                                          members.CheckState
                                                                 := 'l';
                                                          problemflag := 'l';
                                                          end;
                                                  end;

                                    'n'   :   begin
                                                  PutEmTogether;
                                                  Startmerge := True;
                                                  members.CheckState
                                                         := 'o';
                                                  problemflag := 'o';
                                              end;

                                    'p'   :   begin
                                                  PutEmTogether;
                                                  Startmerge := True;

                                                  if changeflag then
                                                      begin
                                                      members.CheckState
                                                             := 'z';
                                                      problemflag := 'z';
                                                      end
                                                  else
                                                      begin
                                                      members.CheckState
                                                             := 'o';
                                                      problemflag := 'o';
                                                      end;
                                              end;

                              end;     {Case Statement}

                       end;    {while statement}

                   end;     {Embedded If Statement}

          close(activeproblemfile);

          if Startmerge then

             begin    {If startmerge Statement}

                Count := 1;

                reset(activeproblemfile);
```

```
              while not EOF(ActiveProblemFile) do

                 begin   {While Statement}

                    read(ActiveProblemFile,members);

                    if (members.Problem = ProbName) and
                       (members.choice = alternative) then
                       begin
                          AllTogether(Names,Limmit);
                          members.CheckState := problemflag;
                       end;

                    seek(activeproblemfile,count-1);
                    write(activeproblemfile,members);
                    count := succ(count);

                 end;   {while statement}

              close(activeproblemfile);

           end;    {If startmerge Statement}

        end;  {if statement}


     end;    {FlagSetter}


  begin   {main program}

     flagsetter;
     (* Returns the screen to normal textmode *)
     textbackground(black);
     textcolor(white);
     clrscr;
     (* Resets the border to black *)
     port[$03d9]:= $f and 0;

  end.    {main program}
```

```
(*****************************************************************)
   FILE        : FILTER1.LIB  (192 lines)
   WRITTEN BY  : Mike Neeley & Bob Wooldridge, May,86
   PURPOSE     : Procedure library for TOUCHSTONE (COOP
                 Criteria Filter Program) written as a part
                 of  a thesis for a Master of Science in
                 Computer   Systems Management, Naval
                 Postgraduate School,   Monterey, California
   CONTENTS    : BASICBOX, CLOSEFILE, SETFILE

(*****************************************************************)
   PROCEDURE   : BASICBOX
   WRITTEN BY  : Mike Neeley & Bob Wooldridge, May,86
                 Based on a program created by Mark Hayes
   PURPOSE     : Draws a box as specified by the input
                 variables
   PARAMETERS  : X1,Y1,X2,Y2 : integers (box corner
                 coordinates)
   EXTERNAL
   NEEDS       : none
(*****************************************************************)

procedure BASICBOX (X1,Y1,X2,Y2:integer);

   var
     BC : array[1..1,1..4] of integer;
     M,I,J : Integer;

begin
                                                {box parameters}
  BC[1,1] := X1;     BC[1,2] := Y1;
  BC[1,3] := X2;     BC[1,4] := Y2;

  for M := 1 to 1 do begin                      {draw a single box as
                                                 needed}
    GotoXY(BC[M,1],BC[M,2]);
    write(chr(201));
    for J := (BC[M,1]+1) to (BC[M,3]-1) do begin
      GotoXY(J,BC[M,2]);
      write(chr(205))
    end;  {for J :=}
    GotoXY(BC[M,3],BC[M,2]);
    write(chr(187));
    for I := (BC[M,2]+1) to (BC[M,4]-1) do begin
      GotoXY(BC[M,1],I);
      write(chr(186));
      GotoXY(BC[M,3],I);
      write(chr(186))
    end;  {for I :=}
    GotoXY(BC[M,1],BC[M,4]);
    write(chr(200));
    for J := (BC[M,1]+1) to (BC[M,3]-1) do begin
      GotoXY(J,BC[M,4]);
      write(chr(205))
    end;  {for J :=}
    GotoXY(BC[M,3],BC[M,4]);
    write(chr(188))
  end; {for M :=}
end;  {procedure BASICBOX}

(************************************************************)
   PROCEDURE   : CLOSEFILE
   WRITTEN BY : Mike Neeley & Bob Wooldridge, May,86
   PURPOSE     : Closes the chatterbox file
   PARAMETERS : none
   EXTERNAL
   NEEDS       : ALTERNATIVE : string[1];
(************************************************************)

procedure CLOSEFILE;

  var
    L, X, COUNTER                         : integer;
    CH                                    : char;
    ANONIMITY                             : char;
    USERFILE                              : string[2];
    PROBLEMNAME                           : string[8];
    CHATRFILE                             : string[14];
    CHECKFILE                             : text;
    CHECKCODE                             : array[1..8] of char;

  procedure GETANS;
    {solits an answer from the user}

    begin
      repeat
        read(kbd,CH);
        if CH in ['a'..'z'] then
          CH := chr(ord(CH)-32);
      until CH in ['A'..'Z',' ',#13];
    end;   {procedure GETANS}


  begin
    clrscr;
    gotoxy (16,5);
    write ('Problem File Name:  *******');
    X := 36; COUNTER := 1;

    repeat    {until COUNTER >8}
      gotoxy(X,5);
      GETANS;
      CHECKCODE[COUNTER] := CH;
      if not(CHECKCODE[1] in [' ',#13]) then begin
        write (CH);
        X := X + 1;
        if (CH = #13) then begin
          CHECKCODE[COUNTER] := ALTERNATIVE;
          for L := (COUNTER +1) to 8 do
            CHECKCODE[L] := ' ';
          COUNTER := 8;
        end;  {if CH=#13}
        if COUNTER = 7 then begin
          CHECKCODE[8] := ALTERNATIVE;
          COUNTER := 8;
        end;  {if COUNTER=7}
        COUNTER := COUNTER + 1;
      end; {if not usercode}
    until (COUNTER > 8);

    PROBLEMNAME := CHECKCODE;

    if PROBLEMNAME <> #13 then begin
      CHATRFILE := concat(FILEDRIVE,':',PROBLEMNAME,'.zzw');
      assign(CHECKFILE,CHATRFILE);
      {$I-}
      reset(CHECKFILE);
      {$I+}
      if IOresult = 0 then begin
        gotoxy (18,10);
        write ('CHATTERBOX FILE CLOSED');
        read(CHECKFILE,USERFILE);
        ANONIMITY := copy(USERFILE,2,1);
        USERFILE := concat('C',ANONIMITY);
        rewrite(CHECKFILE);
        write (CHECKFILE, USERFILE);
      end  {if IOresult}
      else begin
        gotoxy (17,10);
        sound(440);delay(250);nosound;
        write ('CHATTERBOX FILE NOT FOUND');
        delay(1500);
      end;  {else}
      close(CHECKFILE);
      delay(1500);
    end;  {if PROBNAME}
  end;  {procedure CLOSEFILE}

(************************************************************)
   PROCEDURE    : SETFILE (ANONYMOUS:boolean);
   WRITTEN BY : Mike Neeley & Bob Wooldridge, May,86
   PURPOSE      : Sets up the chatterbox file (called in the
                  main program)
   PARAMETERS : ANONYMOUS : boolean;
   EXTERNAL
   NEEDS        : PROBNAME : string[8];
(************************************************************)

procedure SETFILE;

  var
    ANONIMITY                                 : char;
    USERFILE                                  : string[2];
    CHATRFILE                                   : string[14];
    CHECKFILE                                   : text;

  begin
    CHATRFILE := concat(FILEDRIVE,':',PROBNAME,'.zzw');
    assign(CHECKFILE,CHATRFILE);
    if ANONYMOUS then
      ANONIMITY := 'A'
    else
      ANONIMITY := 'N';
    USERFILE := concat('C',ANONIMITY);
    rewrite(CHECKFILE);
    write (CHECKFILE, USERFILE);
    close(CHECKFILE);
  end;  {procedure SETFILE}
```

```
(*****************************************************************)
  FILE        : FILTER2.LIB  (4373)
   WRITTEN BY : Mike Neeley & Bob Wooldridge, May,86
   PURPOSE    : Procedure library for TOUCHSTONE (COOP
Criteria
                Filter Program) written as a part of a
thesis for a
                Master of Science in Computer Systems
Management,
                Naval Postgraduate School, Monterey,
California
   CONTENTS   : CHATRCHECK, SAVESCREEN, WRITESCREEN

(*****************************************************************)
   FUNCTION   : CHATRCHECK
   WRITTEN BY : Mike Neeley & Bob Wooldridge, May,86
   PURPOSE    : Reads the information in two files
                associated with a specific CHATRBOX file and
                determines if the file is being used [and]
                if a recentry entry has been made.
   PARAMETERS : none
   EXTERNAL
   NEEDS      : type
                  STRING3 = string[3];
                  DATASTRING = string [50];
                var
                  ALTERNATIVE : string[1];
(*****************************************************************)

  procedure CHATRCHECK;

    var
      MESSAGEWAITING, CHATAVAILABLE     : boolean;
      CHATRFILE                         : string[14];
      CHECKFILE                         : text;
      USERCHECK                         : char;
      USERNAME                          : string[3];

    begin
      MESSAGEWAITING := false;
      CHATAVAILABLE  := true;

      CHATRFILE :=
      concat(FILEDRIVE,':',PROBNAME,ALTERNATIVE,'.zzw');
      assign (CHECKFILE,CHATRFILE);
      {$I-}
      reset (CHECKFILE);
      {$I+}
      if IOresult = 0 then begin
        read (CHECKFILE,USERCHECK);
        if USERCHECK = 'O' then
          CHATAVAILABLE := false;
      end;  {if IOresult}
    close(CHECKFILE);

    CHATRFILE :=
    concat(FILEDRIVE,':',PROBNAME,ALTERNATIVE,'.zzq');
    assign (CHECKFILE,CHATRFILE);
    {$I-}
    reset (CHECKFILE);
    {$I+}
    if IOresult = 0 then begin
      read (CHECKFILE,USERNAME);
      if USERNAME <> NAMESTRING then
        MESSAGEWAITING := true;
    end;  {if IOresult}
    close(CHECKFILE);

    window(1,1,80,25);
    if not CHATAVAILABLE then begin
      gotoxy (59,25);
      textbackground(red);
      write (' CHATTERBOX IN USE  ');
    end  {if not CHATAVAILABLE}
    else begin
      if MESSAGEWAITING then begin
        gotoxy (59,25);
        textbackground(red);
        write ('NEW CHATTERBOX ENTRY');
      end  {if MESSAGEWAITING}
      else begin
        gotoxy (59,25);
        textbackground(blue);
        write ('CHATTERBOX AVAILABLE');
      end  {if not CHATAVAILABLE}
    end;
    textbackground(blue);
    window(pt1,pt2,pt3,pt4);
end;  {procedure CHATRCHECK}
```

```
(***********************************************************)
   FILE        : FILTER3.LIB  (10579)
   WRITTEN BY : Mike Neeley & Bob Wooldridge, May,86
   PURPOSE     : Procedure library for TOUCHSTONE (COOP
                 Criteria Filter Program) written as a part
                 of a thesis for a Master of Science in
                 Computer Systems Management, Naval
                 Postgraduate School, Monterey, California
   CONTENTS    : SCROLLBOX
(***********************************************************)
  PROCEDURE  : SCROLLBOX
   WRITTEN BY : Mike Neeley & Bob Wooldridge, May,86
                 Word processing section based on a program
                 by Mark Hayes
   PURPOSE     : Reads from a text file, puts text into a
                 specified window, and allows scrolling
                 within that window
   PARAMETERS : XX,YY      : upper left-hand corner of
                             scrollbox
                 ENDTEXT    : length of text (100 lines or
                             less)
                 TITLECODE : designates the title of the
                             scrollbox
   EXTERNAL
   NEEDS       : FILEDRIVE : drive on which the problem
                             elaborator file is located
                 HELPDRIVE : drive on which the help files
                             are located
                 PROBNAME  : name of the textfile called

(***********************************************************)

procedure CHATRBOX
(FILEDRIVE:char;PROBFILE:STRING8;PERSNAME:STRING3);
  forward;

procedure SCROLLBOX
(XX,YY:integer;ENDTEXT:integer;TITLECODE:char);

  type
    WPARRAY = array[1..100,1..50] of char;
    TEXTARRAY = array[1..100] of string[50];
    STRING50 = string[50];

  var
    ENDRUN, USEDFILE, NEWFILE           : boolean;
    SCROLLFOUND, EXPANDFOUND            : boolean;
    TEXTLINE, X, Y, STOP                : integer;
    LASTLINE, ENDSCROLL, CHECKLINE      : integer;
    A,B,F,I,J,K,L,M                     : integer;  {assorted
                                                      counters}
    CH, TEMPCH1, TEMPCH2, USERCHECK     : char;
    INSTRDRIVE                          : char;
  USERNAME, EXTENDER                    : string[3];
  INSTRFILE                             : string[8];
  EXPANDFILE, HELPFILE  : string[14];
  WORKFILE              : text;
  TEMPNAME              : array [1..3] of char;
  TEMPARRAY             : array [1..50] of char;
  NAME                  : array [1..125] of string[3];
  SCREEN                : array[1..25,1..80] of integer;
  ATTRIBUTE             : array[1..25,1..80] of integer;
  WORDPROC              : WPARRAY;
  TEMPLINE, WORDLINE    : TEXTARRAY;


procedure FILLSCREEN (STARTLINE : integer);
    {writes the file array to the screen starting at the}
    {line sent as a parameter}

var
  W,X : integer;

  begin  {procedure FILLSCREEN}
    F := 1;
    if EXPANDFOUND then begin
      for J := STARTLINE to (STARTLINE + 8) do begin
        for K := 1 to 50 do begin
          gotoxy(K,F);
          write (WORDPROC[J,K]);
        end;  {for K}
        F := F + 1;
      end;  {for J}
    end  {if EXPANDFOUND}
    else begin
      for J := STARTLINE to (STARTLINE + 8) do begin
        gotoxy(1,F);
        write (WORDLINE[J]);
        W := wherex;
        for X := W to 50 do
          write(' ');
        F := F + 1;
      end;  {for J}
    end;  {else}
  end;  {procedure FILLSCREEN}


procedure SAVESCREEN(X,Y:integer);
  {Reads the screen under the helpbox into an array}

  begin  {procedure SAVESCREEN}
    for A:= Y to (Y+10) do begin
      for B := X to (X+55) do begin
        SCREEN[A,B] :=
            MemW[$B800:(((A-1)*160)+((B-1)*2))];
        ATTRIBUTE[A,B] :=
            MemW[$B800:(((A-1)*160)+((B-1)*2)+1)];
      end;  {B}
```

```
        end;   {A}
      end;   {procedure SAVESCREEN}


    procedure WRITESCREEN(X,Y:integer);
      {write back the saved portion of the screen}

      begin   {procedure WRITESCREEN}
        for A:= Y to (Y+10) do begin
          for B := X to (X+55) do begin
            MemW[$B800:(((A-1)*160)+((B-1)*2))] :=
                                   SCREEN[A,B];
            MemW[$B800:(((A-1)*160)+((B-1)*2)+1)] :=
                                   ATTRIBUTE[A,B];
          end;  {B}
        end;   {A}
      end;   {procedure WRITESCREEN}


  begin   {procedure SCROLLBOX}
                                 { *** SCREEN SETUP PORTION *** }
    SAVESCREEN(XX,YY);
    textcolor(15); textbackground(2);
    window(1,1,80,25);
    BASICBOX(XX,YY,(XX+55),(YY+10));     {draw SCROLLBOX window
                                               and define}
    textcolor(0); textbackground(15);
    gotoxy ((XX+8),YY);
    case TITLECODE of                      {writes appropriate
                                                title of box}
      'A','a' :
        begin
          write ('          PROBLEM EXPLANATION             ');
          INSTRFILE := PROBNAME;
        end;
      'B','b' :
        begin
          write ('         CHATTERBOX HELP SCREEN           ');
          INSTRFILE := 'CHATRBOX';
        end;
      'C','c' :
        begin
          write (' INSTRUCTION SCREEN #1  (F-10 TO QUIT) ');
          INSTRFILE := 'HELPBX1';
        end;
      'D','d' :
        begin
          write (' INSTRUCTION SCREEN #2  (F-10 TO QUIT) ');
          INSTRFILE := 'HELPBX2';
        end;
      'E','e' :
        begin
          write (' INSTRUCTION SCREEN #3  (F-10 TO QUIT) ');
          INSTRFILE := 'HELPBX3';
        end;
```

```
'F','f' :
  begin
    write (' INSTRUCTION SCREEN #4  (F-10 TO QUIT) ');
    INSTRFILE := 'HELPBX4';
  end;
'G','g' :
  begin
    write (' INSTRUCTION SCREEN #5  (F-10 TO QUIT) ');
    INSTRFILE := 'HELPBX5';
  end;
'H','h' :
  begin
    write (' INSTRUCTION SCREEN #6  (F-10 TO QUIT) ');
    INSTRFILE := 'HELPBX6';
  end;
'I','i' :
  begin
    write (' INSTRUCTION SCREEN #7  (F-10 TO QUIT) ');
    INSTRFILE := 'HELPBX7';
  end;
'J','j' :
  begin
    write (' INSTRUCTION SCREEN #8  (F-10 TO QUIT) ');
    INSTRFILE := 'HELPBX8';
  end;
'K','k' :
  begin
    write (' INSTRUCTION SCREEN #9  (F-10 TO QUIT) ');
    INSTRFILE := 'HELPBX9';
  end;
'L','l' :
  begin
    write (' INSTRUCTION SCREEN #10 (F-10 TO QUIT) ');
    INSTRFILE := 'HELPBX10';
  end;
'M','m' :
  begin
    write (' INSTRUCTION SCREEN #11 (F-10 TO QUIT) ');
    INSTRFILE := 'HELPBX11';
  end;
'N','n' :
  begin
    write (' INSTRUCTION SCREEN #12 (F-10 TO QUIT) ');
    INSTRFILE := 'HELPBX12';
  end;
'O','o' :
  begin
    write (' INSTRUCTION SCREEN #13 (F-10 TO QUIT) ');
    INSTRFILE := 'HELPBX13';
  end;
'P','p' :
  begin
    write (' INSTRUCTION SCREEN #14 (F-10 TO QUIT) ');
    INSTRFILE := 'HELPBX14';
  end;
```

```
    'Z','z' :
      begin
        write ('          PROBLEM EXPLANATION              ');
      end;
  end;  {case TITLECODE}

  if TITLECODE in ['A','a','Z','z'] then begin
    EXTENDER := 'zzx';
    INSTRDRIVE := FILEDRIVE;
  end   {if TITLECODE}
  else begin
    EXTENDER := 'zzy';
    INSTRDRIVE := HELPDRIVE;
  end;  {else}

  if TITLECODE in ['Z','z'] then begin
    gotoxy ((XX),(YY+10));
    write (' USE:  UP&DN ARROW KEYS,
               HOME,END,PG UP,PG DN,F-10(quit) ');
  end  {if TITLECODE}
  else begin
    gotoxy ((XX+2),(YY+10));
    write ('USE:  ARROW KEYS,HOME,END,PG UP,PG
                DN,TAB,DEL,RETURN');
  end; {else}
  textcolor(15); textbackground(2);
  window((XX+1),(YY+1),(XX+54),(YY+9));{clear the screen
                                        area}
  clrscr;
  window((XX+3),(YY+1),(XX+53),(YY+9));{identify the text
                                         parameters}


                    { *** FILE SETUP PORTION *** }

  EXPANDFOUND := false;
  if TITLECODE in ['Z','z'] then begin
    for J := 1 to 50 do
      for K := 1 to 100 do                {the wordprocessing
                                             array is}
        WORDPROC[K,J] := chr(32);         {initialized to all
                                             blanks}
    EXPANDFOUND := true
  end  {if TITLECODE}
  else begin
    HELPFILE :=
concat(INSTRDRIVE,':',INSTRFILE,'.',EXTENDER);
    assign (WORKFILE,HELPFILE);          {open CHATRFILE and
                                            read into}
    {$I-}
    reset (WORKFILE);                    {word processing
                                            array}
    {$I+}
    if IOresult = 0 then begin
      I := 1;
    while (not eof(WORKFILE)) and (I <(ENDTEXT+1)) do
  begin
      readln (WORKFILE,WORDLINE[I]);
      I := I + 1;
    end;  {while not eof}
    SCROLLFOUND := true;
  end  {if IOresult}
  else begin
    gotoxy(5,5);
    if TITLECODE in ['A','a'] then
      write ('NO EXTENDED EXPLANATION FOR THIS PROBLEM')
    else
      write ('       HELP FILE NOT FOUND ON DISK');
    gotoxy(5,7);
    write ('       Press any key to continue ');
    SCROLLFOUND := false;
    repeat until keypressed;
  end;  {else}
  close (WORKFILE);
end;  {else}
                 { *** INITIALIZATION *** }


LASTLINE := 1;
USEDFILE := false;     {of run & the "dirty bit" flag}

if SCROLLFOUND or EXPANDFOUND then begin
  FILLSCREEN (1);
  if EXPANDFOUND then
    ENDSCROLL := 100
  else
    ENDSCROLL := ENDTEXT; {designate last line of scroll-}
                          {only text}
  X := 1; Y:= 1;          {initialize column, row}
  TEXTLINE := 1;          {initialize textline}
  ENDRUN := false;      {initialize flag for end of run}

                 { *** SCROLLING ROUTINE *** }

  repeat
    gotoxy(X,Y);
    read (kbd,CH);
    case CH of
      #32..#126 :     {regular characters}
          if EXPANDFOUND then begin
            USEDFILE := true;
            if WORDPROC[TEXTLINE,X] <> chr(32) then begin
              TEMPCH1 := CH;
              for K := X to 50 do begin
                TEMPCH2 := WORDPROC[TEXTLINE,K];
                WORDPROC[TEXTLINE,K] := TEMPCH1;
                gotoxy(K,TEXTLINE);
                write(WORDPROC[TEXTLINE,K]);
                TEMPCH1 := TEMPCH2;
              end;  {for K=X to 50}
          end   {if WORDPROC <> chr(32)}
          else begin
            WORDPROC[TEXTLINE,X] := ch;
            write(ch);
          end;   {else}
          X := X + 1;
          if TEXTLINE > LASTLINE then
            LASTLINE := TEXTLINE;
        end;  {if EXPANDFOUND}
    #8:                                       {BACKSPACE}
        if EXPANDFOUND then begin
          if X > 1 then begin
            X := X - 1;
            gotoxy(X,Y);
            for J := X to 49 do begin;
              WORDPROC[TEXTLINE,J] :=
                WORDPROC[TEXTLINE,J+1];
              write(WORDPROC[TEXTLINE,J]);
            end;  {for}
          end;  {if X>1}
          WORDPROC[TEXTLINE,50] := chr(32);
          gotoxy(50,Y);
          write(WORDPROC[TEXTLINE,50]);
        end;  {if EXPANDFOUND}
    #9 :                                      {TAB key}
        if EXPANDFOUND then
          if X < 46 then
            X := X + 5;
    #13:                                      {RETURN key}
        if EXPANDFOUND then begin
          if Y < 9 then begin
            Y := Y + 1;
            X := 1;
            TEXTLINE := TEXTLINE + 1:
          end  {if Y<9}
          else begin
            if TEXTLINE <100 then begin
              TEXTLINE := TEXTLINE + 1;
              FILLSCREEN(TEXTLINE - 8);
              X := 1;
            end  {if TEXTLINE <100}
            else begin
              sound(800);delay(50);nosound;  {Too far
                                                    down}
            end; {else}
          end;  {else}
        end;  {if EXPANDFOUND}
    #27:                                     {ESCAPE key}
      begin
        read (kbd,CH);
        case CH of
(*        #60 :
            begin
              if not (TITLECODE in ['B','b']) then begin
                CHATRBOX(FILEDRIVE,PROBNAME,NAMESTRING);
```

```
                 window((XX+3),(YY+1),(XX+53),(YY+9));
                 textcolor(15); textbackground(2);
               end;  {if not TITLECODE}
             end;  {case #60}
*)         #68 : ENDRUN := true;       {Key F-10 to quit }
           #72 :                       {UP arrow key}
             begin
               if Y > 1 then begin
                 Y := Y - 1;
                 TEXTLINE := TEXTLINE - 1;
               end  {if Y>1}
               else begin
                 if TEXTLINE > 1 then begin
                   TEXTLINE := TEXTLINE - 1;
                   FILLSCREEN (TEXTLINE);
                 end   {if TEXTLINE>1}
                 else begin
                   sound(2800);delay(50);nosound;
                     {Too far up}
                 end;  {else}
               end;  {else}
             end;  {#72 case}
           #80 :                       {DOWN arrow key}
             begin
               if Y < 9 then begin
                 Y := Y + 1;
                 TEXTLINE := TEXTLINE + 1;
               end  {if}
               else begin
                 if TEXTLINE < ENDSCROLL then begin
                   TEXTLINE := TEXTLINE + 1;
                   FILLSCREEN (TEXTLINE - 8);
                 end  {if TEXTLINE<ENDSCROLL}
                 else begin
                   sound(800);delay(50);nosound; {Too far
                                                   down}
                 end;  {else}
               end;  {else}
             end;   {#80 case}
           #77 :                       {RIGHT arrow key}
               if EXPANDFOUND then begin
                 if X < 50 then
                   X := X + 1
                 else begin
                   sound(2000);delay(50);nosound; {Too
                                              far right}
                 end;  {else}
               end;  {if EXPANDFOUND}
           #75 :                       {LEFT arrow key}
               if EXPANDFOUND then begin
                 if X > 1 then
                   X := X - 1
                 else begin
                   sound(1200);delay(50);nosound; {Too
                                              far left}
```

```
        end;  {else}
      end;  {else}
  #83 :                              {DELETE key}
      if EXPANDFOUND then begin
        for J := X to 49 do begin
          WORDPROC[TEXTLINE,J] :=
             WORDPROC[TEXTLINE,J+1];
          write(WORDPROC[TEXTLINE,J]);
        end;   {for}
        WORDPROC[TEXTLINE,50] := chr(32);
        gotoxy(50,Y);
        write(WORDPROC[TEXTLINE,50]);
      end;  {if EXPANDFOUND}
  #73 :                              {PG UP key}
     begin
      if (TEXTLINE-Y) > 9 then
        TEXTLINE := TEXTLINE - 9
      else begin
        sound(2800);delay(50);nosound;  {Too far
                                                up}
        TEXTLINE := 1;
        Y := 1;
      end;  {else}
      FILLSCREEN(TEXTLINE-Y+1);
    end;  {#73 case}
  #81 :                              {PG DN key}
    begin
      if EXPANDFOUND then begin
        if TEXTLINE < 91 then
          TEXTLINE := TEXTLINE + 9
        else begin
          sound(800);delay(50);nosound;  {Too far
                                                down}
          TEXTLINE := 100;
          Y := 9;
        end;  {else}
      end  {if EXPANDFOUND}
      else begin
        if (TEXTLINE - Y) < (ENDSCROLL-18) then
          TEXTLINE := TEXTLINE + 9
        else begin
          sound(800);delay(50);nosound;  {Too far
                                                down}
          TEXTLINE := ENDSCROLL;
          Y := 9;
        end;  {else}
      end;  {else}
      FILLSCREEN(TEXTLINE - Y + 1);
    end;  {#81 case}
  #71 :                              {HOME key}
    begin
      TEXTLINE := 1;
      X := 1; Y := 1;
      FILLSCREEN (TEXTLINE);
    end;   {#71 case}
```

```
                #79 :                               {END key}
                    if EXPANDFOUND then begin
                      TEXTLINE := LASTLINE;
                      if LASTLINE > 9 then begin
                        Y := 9;
                        FILLSCREEN(TEXTLINE - 8);
                      end  {if LASTLINE>9}
                      else begin
                        FILLSCREEN(1);
                        Y := LASTLINE;
                      end;  {else}
                      X := 50;
                    end  {if EXPANDFOUND}
                    else begin
                      TEXTLINE := ENDSCROLL;
                      Y := 9;
                      FILLSCREEN (ENDSCROLL-8);
                    end;  {else}
              end;  {case CH of}
            end;  {#27}
        end;  {case CH of}

* * * WORD WRAP PORTION OF THE WORDPROCESSING ROUTINE * * *
* At the end of the line, if the user is still typing,    *
* this section causes the line to wrap around to the next *
* line.                                                   *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

        if (X > 50) and (TEXTLINE < 100) and EXPANDFOUND then
       begin
          X := 50;
          if WORDPROC[TEXTLINE,X] <> chr(32) then begin
                      {test last char in line for}
                      {blank}
            while WORDPROC[TEXTLINE,X] <> chr(32) do
            X := X - 1;                        {reset X to pos of
                                                last blank  }
            for M := (X + 1) to 50 do begin
              WORDPROC[TEXTLINE+1,M-X] :=
                     WORDPROC[TEXTLINE,M];
                            {move the char to correct }
              gotoxy(M,Y);  {array pos}
              WORDPROC[TEXTLINE,M] := chr(32);
              write(WORDPROC[TEXTLINE,M]); {erase word from
                                            end of line}
              if Y < 9 then begin
                gotoxy(M-X,Y+1);
                write(WORDPROC[TEXTLINE+1,M-X]);
               {write word at front of new line}
              end; {if}
            end;   {for}
            X := (M-X) +1;
          end   {if}
          else
            X := 1;
          TEXTLINE := TEXTLINE + 1;

          if Y < 9 then
            Y := Y + 1
        else begin
          Y := 3;
          FILLSCREEN (TEXTLINE - 2);
          sound(2000);delay(50);nosound;
          sound(800);delay(50);nosound;
          sound(1200);delay(50);nosound;
          sound(2000);delay(50);nosound;
        end;  {else}
      end  {if}
    until ENDRUN;

(* * * SAVE FILE PORTION * * * * * * * * * * * * * * * * * * * *
* At the end of the wordprocessing session, the file is   *
* saved by moving all text lines to the end of the file   *
* so they can be readjusted when the file is next opened. *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *)

    if USEDFILE then begin
      clrscr;
      textcolor(15); textbackground(0);
      gotoxy(11,5);
      write(' SAVING PROBLEM EXPLANATION ');
      EXPANDFILE := concat(FILEDRIVE,':',PROBNAME,'.zzx');
      assign (WORKFILE,EXPANDFILE);          {open CHATRFILE
                                               and read into }
      rewrite (WORKFILE);     {save the array to disk}
      for J := 1 to 50 do begin
        for K := 1 to 50 do
          TEMPARRAY[K] := WORDPROC[J,K];
        TEMPLINE[J] := TEMPARRAY;
      end;  {for J := 1 to 50}
      for J := 1 to 50 do begin
        writeln (WORKFILE,TEMPLINE[J]);
      end;  {for J}
      close(WORKFILE);
    end;  {if USEDFILE}

  end;  {if SCROLLFOUND or EXPANDFOUND}

* * * RETURN TO NORMAL ROUTINE * * * * * * * * * * * * * *
* Clears the chatterbox window and rewrites the screen    *
* portion that was saved when chatterbox was invoked.     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *

  window (XX,YY,(XX+55),(YY+10));
  clrscr;
  window (1,1,80,25);
  WRITESCREEN (XX,YY);                   {write previous
                                          screen back}
end;   {procedure SCROLLBOX}
```

```
(**************************************************************)
 FILE        : FILTER4.LIB  (26092)
  WRITTEN BY : Mike Neeley & Bob Wooldridge, May,86
  PURPOSE    : Procedure library for TOUCHSTONE (COOP
               Criteria Filter Program) written as a part
               of a thesis for a Master of Science in
               Computer Systems Management, Naval
               Postgraduate School, Monterey, California
  CONTENTS   : CHATRBOX
(**************************************************************)
  PROCEDURE  : CHATRBOX
  WRITTEN BY : Mike Neeley & Bob Wooldridge, May,86
               Word processing section based on a program
               by Mark Hayes
  PURPOSE    : Reads from a text file, puts text into a
               specified window, and allows scrolling
               within that window for previous input;
               allows word processing for the new input.
  PARAMETERS : FILEDRIVE    : drive on which the file is
                              located
               PROBFILE     : name of the textfile called
               PRESNAME     : name/initials of the user
               ALTERNATIVE  : Character designating
                              criteria/alternative selection
  EXTERNAL
  NEEDS      : type
                 STRING8 = string[8]
                 STRING3 = string[3]
               var
                 ALTERNATIVE : string[1];
               INCLUDE file  :
FILTER1.LIB,FILTER2.LIB,FILTER3.LIB
(**************************************************************)

procedure CHATRBOX
{(FILEDRIVE:char;PROBFILE:STRING8;PERSNAME:STRING3)};

  type
    WPARRAY = array[1..125,1..55] of char;
    FILEARRAY = array[1..80] of string[55];
    DATASTRING = STRING[50];
  var
    SCREEN                              : array[1..25,1..80]
of integer;
    ATTRIBUTE                           : array[1..25,1..80]
of integer;
    ENDRUN, USEDFILE, NEWENTRYSEEN      : boolean;
    CHANGE, SCROLLONLY,OKAY_TO_CHAT     : boolean;
    NEWFILE, NEWLINE                    : boolean;
    TEXTLINE, X, Y, STOP                : integer;
    LASTLINE, ENDSCROLL, CHECKLINE      : integer;
    A,B,F,I,J,K,L,M                     : integer;  {assorted
                                                     counters}
```

```
      CH, TEMPCH1, TEMPCH2, USERCHECK    : char;
      ANONIMITY                    : char;
      USERFILE                     : string[2];
      USERNAME                     : string[3];
      CHATRFILE, SAVEFILE          : string[14];
      TEMPLINE                     : string[55];
      BUFFERLINE                   : string[55];
      TEMPNAME                     : array [1..3] of char;
      DATELINE,TEMPARRAY           : array [1..55] of char;
      NAME                         : array [1..125] of string[3];
      CHECKFILE, TEXTFILE          : text;
      SAVELINE                     : FILEARRAY;
      WORDPROC                     : WPARRAY;

    function WRITEDATE(PERSNAME : STRING3): DATASTRING;
  {writes the date/time on a line at the bottom of an entry}

      type
        REGISTERS = record
                      AX,BX,CX,DX,BP,SI,DS,ES,FLAGS : integer;
                    end;    {record}

      var
        REGS                                  : REGISTERS;
        STRDATE                               : string[10];
        STRTIME                               : string[5];
        DA,MO,HR,MN                           : string[2];
        YR                                    : string[4];
        I                                     : integer;

      begin  {function WRITEDATE}
        with REGS do begin
          AX := $2A00;
          msdos(REGS);
          str(CX,YR);
          str(lo(DX),DA);
          if (lo(DX) < 10) then
            DA := concat('0',DA);
          str(hi(DX),MO);
          if (hi(DX) < 10) then
            MO := concat('0',MO);
        end;  {with REGS}
        with REGS do begin
          AX := $2C00;
          msdos(REGS);
          str(hi(CX),HR);
          str(lo(CX),MN);
          if (lo(CX) < 10) then
            MN := concat('0',MN);
        end;
        if LENGTH(PERSNAME)<3 then
          for I := 1 to (3-LENGTH(PERSNAME)) do
            PERSNAME := concat(PERSNAME,' ');
      if NEWLINE then begin
        WRITEDATE := concat('**** ',PROBFILE,' FILE BEGUN:
                        ',MO,'/',DA,
          '/',YR,' @ ',HR,':',MN,' **********');
        NEWLINE := false;
      end   {if NEWLINE}
      else
        WRITEDATE := concat('** MESSAGE ENDED:
                        ',MO,'/',DA,'/',YR,
          ' @ ',HR,':',MN,' **** ',PERSNAME,' ****');
    end;  {function WRITEDATE}

  procedure FILLSCREEN (STARTLINE : integer);
{Sub-procedure to write the array to the screen starting
       at the line number sent as a parameter}

    begin  {procedure FILLSCREEN}
      F := 3;
      for J := STARTLINE to (STARTLINE + 6) do begin
        if (WORDPROC[J,2]='*') and (WORDPROC[J,48]='*')
          and (WORDPROC[J,49]='*') then textbackground(1)
        else
          textbackground(4);
        for K := 1 to 50 do begin
          gotoxy(K,F);
          write(WORDPROC[J,K]);
        end;  {for K := 1 to 50}
        F := F + 1;
      end;  {for J}
      textbackground(4);
    end;  {procedure FILLSCREEN}

  procedure SAVESCREEN(X,Y:integer);
    {Reads the screen under the helpbox into an array}

    begin  {procedure SAVESCREEN}
      for A:= Y to (Y+10) do begin
        for B := X to (X+55) do begin
          SCREEN[A,B] :=
         MemW[$B800:(((A-1)*160)+((B-1)*2))];
          ATTRIBUTE[A,B] :=
         MemW[$B800:(((A-1)*160)+((B-1)*2)+1)];
        end;  {B}
      end;  {A}
    end;  {procedure SAVESCREEN}
  procedure WRITESCREEN(X,Y:integer);
    {write back the saved portion of the screen}

    begin  {procedure WRITESCREEN}
      for A:= Y to (Y+10) do begin
        for B := X to (X+55) do begin
          MemW[$B800:(((A-1)*160)+((B-1)*2))] :=
                    SCREEN[A,B];
          MemW[$B800:(((A-1)*160)+((B-1)*2)+1)] :=
                   ATTRIBUTE[A,B];
        end;  {B}
      end;  {A}
    end;  {procedure WRITESCREEN}

begin  {procedure CHATRBOX}

(* * * SCREEN SET-UP PORTION * * * * * * * * * * * * * * * *
* Saves the protion of the screen under the chatterbox,   *
* and initializes the color and size of the chatterbox.   *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *)

  SAVESCREEN(23,12);
  textcolor(15); textbackground(4);
  window(1,1,80,25);
  BASICBOX(23,12,78,22);                  {draw CHATRBOX window
                                           and define}
  textcolor(0);textbackground(15);
  gotoxy (28,12);
  write (' CHATTERBOX    [F-1 for help, F-10 to quit] ');
  gotoxy (25,22);
  write ('USE:  ARROW KEYS,HOME,END,PG UP,PG
              DN,TAB,DEL,RETURN');
  textcolor(15); textbackground(4);
  window(24,13,77,21);    {the parameters of the text}
  clrscr;                 {manipulation area and clear the}
  window(26,13,76,21);    {screen in that sector}
  gotoxy(40,1);
  write ('LINE #:');

* * * CHATTERBOX-IN-USE CHECK * * * * * * * * * * * * * * *
* Checks to see if chatterbox is in use; if so, puts a    *
* message on the screen to say so.                        *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *

  NEWFILE := false;
  CHATRFILE :=
     concat(FILEDRIVE,':',PROBFILE,ALTERNATIVE,'.zzw');
  assign(CHECKFILE,CHATRFILE);
  {$I-}                  {Turn I/O off, check for the    }
  reset(CHECKFILE);      {existence of the chatrfile, &  }
  {$I+}                  {turn the I/O back on           }
  if IOresult = 0 then begin
    read(CHECKFILE,USERFILE);
    USERCHECK := copy(USERFILE,1,1);
    ANONIMITY := copy(USERFILE,2,1);
```

```
      if (USERCHECK = 'C') then
         begin    {If chatterbox is being used,   }
        OKAY_TO_CHAT := true;   {the "zzw" file will contain
an }
        USERFILE := concat('O',ANONIMITY);
        rewrite(CHECKFILE);  {"O" for open; otherwise, it    }
        write(CHECKFILE,USERFILE);      {will have a "C" for
                                          closed.    }
      end
      else
        OKAY_TO_CHAT := false;
    end
    else begin
      OKAY_TO_CHAT := true;
      USERFILE := concat('O',ANONIMITY);
      rewrite(CHECKFILE);
      write(CHECKFILE,USERFILE);
      NEWFILE := true;
    end;
    close(CHECKFILE);

* * * FILE SET-UP PORTION * * * * * * * * * * * * * * * * * *
* Gets and sets up the chatterbox file; moves all info    *
* in file to first 80 lines and clears last 45 lines;     *
* no information before the last 80 lines is displayed but*
* all information is maintained in the chatterbox file.   *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *

    NEWLINE := false;
    for J := 1 to 55 do       {At the beginning of the pro- }
      for K := 1 to 125 do   {gram,the wordprocessing array}
        WORDPROC[K,J] := chr(32); {is initialized to all
                                    blanks }

    SAVEFILE :=
       concat(FILEDRIVE,':',PROBFILE,ALTERNATIVE,'.zzz');
    assign (TEXTFILE,SAVEFILE);{open CHATRFILE and read into}
                              {word processing array        }
    {$I-}                     {Turn I/O off, check for the  }
    reset(TEXTFILE);          {existence of the workfile, & }
    {$I+}                     {turn the I/O back on         }
    if IOresult = 0 then begin
      I := 1;
      while not eof (TEXTFILE) do begin
        readln(TEXTFILE);     {This section counts the      }
        I := I + 1;           {number of lines in the text  }
      end;  {while not eof}   {file and uses that informa-  }
      I := I - 1;             {tion to put the correct info }
      if I > 80 then begin    {into the array.  If more than}
        K := I - 80;          {80 lines are in the file,    }
        M := 0;               {only the last 80 are put into}
        reset (TEXTFILE);     {chatterbox for review.       }
        for J := 1 to K do
          readln(TEXTFILE);
      end  {if I>80}
      else begin
        K := 1;
        M := 80-I;
        reset (TEXTFILE);
      end;
      for L := K to I do
        readln (TEXTFILE,SAVELINE[(L-K) + 1 + M]);
      for I := (M+1) to 80 do begin
        BUFFERLINE := SAVELINE[I];
        for J := 1 to 55 do
          WORDPROC[I,J] := BUFFERLINE[J];
      end;  {for I := 1}
    end  {if IOresult}
    else
      NEWLINE := true;
    close (TEXTFILE);

  * * * LOCATE PREVIOUS FILES * * * * * * * * * * * * * * * * *
  * Locates last incident of user name and starts file        *
  * review at that point; if last incident is after line 77,*
  * the file review starts at line 82.                         *
  * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

    for J := 1 to 80 do
      for I := 51 to 53 do begin
        TEMPNAME[I-50] := WORDPROC[J,I];
        NAME[J] := TEMPNAME;
      end;
    USERNAME := PERSNAME;

    for I := 1 to 3 do        {Change username to all caps  }
      if USERNAME[I] in ['a'..'z'] then
        USERNAME[I] := chr(ord(USERNAME[I]) - 32);

    if LENGTH(USERNAME)<3 then
      for I := 1 to (3-LENGTH(USERNAME)) do
        USERNAME := concat(USERNAME,' ');
    I := 80;
    if not(NEWFILE) then
      while (I > 1) and (NAME[I] <> USERNAME) and (NAME[I] <>
  'ZZZ') do
        I := I - 1
    else begin
      TEMPNAME := 'ZZZ';          {Set up "file begin" line}
      for J := 51 to 53 do
        WORDPROC[81,J] := TEMPNAME[J-50];
      TEMPLINE := WRITEDATE ('***');
      for J := 2 to 49 do
        WORDPROC[81,J] := TEMPLINE[J];
    end;  {else}
```

```
* * * INITIALIZATION * * * * * * * * * * * * * * * * * * *
* Initializes all the necessary valuables needed to start *
* the word-processing/scrolling section of the procedure. *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

  if I > 76 then begin
    SCROLLONLY := true;
    FILLSCREEN (76);
    TEXTLINE := 82;
  end  {if I>79}
  else begin
    SCROLLONLY := false;  {initialize flags for info line}
    FILLSCREEN (I);       {draw initial screen          }
    TEXTLINE := I + 6;
  end;  {else}
  if I <> 80 then begin   {indicates on the screen when }
    textcolor(31); textbackground(4);  {there are new
                                  entries not yet}
    gotoxy(25,1);         {seen/answered by the user    }
    write ('*NEW ENTRIES*');
    textcolor(15); textbackground(4);
    NEWENTRYSEEN := false;
  end  {if I<>80}
  else NEWENTRYSEEN := true;
  CHANGE := true;
  ENDSCROLL := 81;        {designate last line of scroll}
                          {only text                    }
  X := 1; Y:= 9;          {initialize column, row, and  }
  LASTLINE := 81;         {last line of text            }
  ENDRUN := false;        {initialize the flag for end  }
  USEDFILE := false;      {of run & the "dirty bit" flag}
  CHECKLINE := 0;

* * * WORDPROCESSING ROUTINE * * * * * * * * * * * * * * *
* A simple wordprocessor which allows the user to write   *
* messages in the chatterbox.                             *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

  repeat               {begin wordprocessing routine   }
    if CHECKLINE <> TEXTLINE then begin
      gotoxy (49,1);
      write ('  ');
      gotoxy(48,1); {write text line on top of       }
      write(TEXTLINE);      {chatterbox              }
      CHECKLINE := TEXTLINE;
      if (TEXTLINE <= ENDSCROLL) and
          (SCROLLONLY = false) then
        CHANGE := true;
      if (TEXTLINE > ENDSCROLL) and (SCROLLONLY = true) then
        CHANGE := true;
      if CHANGE then begin      {this section tests for a}
        gotoxy (1,1);           {change in the status     }

      if SCROLLONLY then
         begin                {from wordprocessing to  }
```

```
          textcolor(1);
          textbackground(12);  {scroll-only and changes }
          write ('WORDPROCESSING SECTION');
            {the label in the cht/box}
        SCROLLONLY := false;
        if NEWENTRYSEEN = false then begin   {erases NEW
                                                  ENTRIES    }
          textcolor(15); textbackground(4);  {flag on
                                                  screen}
          gotoxy(25,1);
          write ('              ');
          NEWENTRYSEEN := true;
        end;    {if NEWENTRYSEEN=true}
      end  {if SCROLLONLY}
      else begin
        textcolor(0); textbackground(10);
        write ('SCROLLING-ONLY SECTION');
        SCROLLONLY := true;
      end;  {else}
      textcolor(15); textbackground(4);
      sound(1840);delay(100);nosound;
      CHANGE := false;
    end;  {if CHANGE}
  end;   {if CHECKLINE}
  gotoxy(X,Y);
  read (kbd,CH);
  case CH of
    #32..#126 :                     {regular characters}
      begin
        if (TEXTLINE > ENDSCROLL) and OKAY_TO_CHAT then
        begin
          USEDFILE := true;
          if WORDPROC[TEXTLINE,X] <> chr(32) then begin
            TEMPCH1 := CH;
            for K := X to 50 do begin
              TEMPCH2 := WORDPROC[TEXTLINE,K];
              WORDPROC[TEXTLINE,K] := TEMPCH1;
              gotoxy(K,TEXTLINE);
              write(WORDPROC[TEXTLINE,K]);
              TEMPCH1 := TEMPCH2;
            end;  {for K=X to 50}
          end   {if WORDPROC <> chr(32)}
          else begin
            WORDPROC[TEXTLINE,X] := ch;
            write(ch);
          end;   {else}
          X := X + 1;
          if TEXTLINE > LASTLINE then
            LASTLINE := TEXTLINE;
        end;     {if TEXTFILE > ENDSCROLL}
        if (TEXTLINE > ENDSCROLL) and not (OKAY_TO_CHAT)
       then begin
          clrscr;
          textcolor(31); textbackground(0);
          gotoxy(6,4);
```

```
            write(' CHATTERBOX IN USE - Word processing   ');
            gotoxy(6,5);
            write('     not available at this time.       ');
            gotoxy(6,6);
            write('      Press any key to continue        ');
            textcolor(15); textbackground(4);
            repeat until keypressed;
            gotoxy (1,1);
            textcolor(1); textbackground(12);
            write ('WORDPROCESSING SECTION');
            textcolor(15); textbackground(4);
            gotoxy(40,1);
            write ('LINE #: 82');
            TEXTLINE := 82;
            Y := 9;
            FILLSCREEN (TEXTLINE - 6);
          end;    {if (TEXTLINE>ENDSCROLL)}
        end;    {case #32-#126}
      #8:                                         {BACKSPACE}
        begin
          if X > 1 then begin
            X := X - 1;
            gotoxy(X,Y);
            for J := X to 49 do begin;
              WORDPROC[TEXTLINE,J] :=
                   WORDPROC[TEXTLINE,J+1];
              write(WORDPROC[TEXTLINE,J]);
            end;  {for}
          end;  {if X>1}
          WORDPROC[TEXTLINE,50] := chr(32);
          gotoxy(50,Y);
          write(WORDPROC[TEXTLINE,50]);
        end;  {case #8}
      #9 :                                        {TAB key}
        begin
          X := 30;
        end;  {case #9}
      #13:                                        {RETURN key}
        begin
          if Y < 9 then begin
            Y := Y + 1;
            X := 1;
            TEXTLINE := TEXTLINE + 1;
          end  {if Y<9}
          else begin
            if TEXTLINE <123 then begin
              TEXTLINE := TEXTLINE + 1;
              FILLSCREEN(TEXTLINE - 6);
              X := 1;
            end  {if TEXTLINE <123}
            else
              sound(800);delay(50);nosound; {Too far down}
          end;  {else}
          if TEXTLINE > LASTLINE then
            LASTLINE := TEXTLINE;
```

```
  end;    {#13 case}
#27:                                    {ESCAPE key}
  begin
  read (kbd,CH);
    case CH of
      #59 :
        begin                    {F-1 key for help}
          SCROLLBOX(8,4,53,'B');
          textcolor(15); textbackground(4);
          window(26,13,76,21);
{NOTE:  after scrollbox is called, color and
        window must be reinitialized by the
        originating program}
        end;
      #61 :
        begin
          SCROLLBOX (4,4,50,'A');
          textcolor(15); textbackground(4);
          window(26,13,76,21);
{NOTE:  after scrollbox is called, color and
        window must be reinitialized by the
        originating program}
        end;
      #68 : ENDRUN := true;       {Key F-10 to quit }
      #72 :                       {UP arrow key}
        begin
          if Y > 3 then begin
            Y := Y - 1;
            TEXTLINE := TEXTLINE - 1;
          end  {if Y>1}
          else begin
            if TEXTLINE > 1 then begin
              TEXTLINE := TEXTLINE - 1;
              FILLSCREEN (TEXTLINE);
            end   {if TEXTLINE>1}
            else
              sound(2800);delay(50);
              nosound; {Too far up}
          end;  {else}
        end;  {#72 case}
      #80 :                       {DOWN arrow key}
        begin
          if Y < 9 then begin
            Y := Y + 1;
            TEXTLINE := TEXTLINE + 1;
          end  {if}
          else begin
            if TEXTLINE < 123 then begin
              TEXTLINE := TEXTLINE + 1;
              FILLSCREEN (TEXTLINE - 6);
            end  {if TEXTLINE<123}
            else
              sound(800);delay(50);
              nosound; {Too far down}
          end;  {else}
      if TEXTLINE > LASTLINE then
        LASTLINE := TEXTLINE;
    end;    {#80 case}
  #77 :                               {RIGHT arrow key}
    begin
      if X < 50 then begin
        X := X + 1;
      end   {if}
      else
        sound(2000);delay(50);
        nosound; {Too far right}
    end;  {#77 case}
  #75 :                               {LEFT arrow key}
    begin
      if X > 1 then begin
        X := X - 1;
      end  {if}
      else
        sound(1200);delay(50);
        nosound; {Too far left}
    end;  {#75 case}
  #83 :                               {DELETE key}
    begin
      if TEXTLINE > ENDSCROLL then begin
        for J := X to 49 do begin
          WORDPROC[TEXTLINE,J] :=
               WORDPROC[TEXTLINE,J+1];
          write(WORDPROC[TEXTLINE,J]);
        end;    {for}
        WORDPROC[TEXTLINE,50] := chr(32);
        gotoxy(50,Y);
        write(WORDPROC[TEXTLINE,50]);
      end;   {if TEXTLINE>ENDSCROLL}
    end;  {#83 case}
  #73 :                               {PG UP key}
    begin
      if TEXTLINE > 7 then
        TEXTLINE := TEXTLINE - 7
      else begin
        sound(2800);delay(50);
        nosound; {Too far up}
        TEXTLINE := 1;
        Y := 3;
      end;  {else}
      FILLSCREEN(TEXTLINE - Y + 3);
    end;  {#73 case}
  #81 :                               {PG DN key}
    begin
      if USEDFILE or NEWENTRYSEEN then
        STOP := 123 else STOP := 82;
      if TEXTLINE < (STOP-7) then
        TEXTLINE := TEXTLINE + 7
      else begin
```

```
                if STOP = 123 then begin
                  sound(800);delay(50);
                  nosound; {Too far down}
                end;   {if STOP=123}
                TEXTLINE := STOP;
                Y := 9;
              end;  {else}
              FILLSCREEN(TEXTLINE - Y + 3);
            end;   {#81 case}
          #71 :                         {HOME key}
            begin
              TEXTLINE := 1;
              Y := 3;
              FILLSCREEN (TEXTLINE);
            end;    {#71 case}
          #79 :                         {END key}
            begin
              if USEDFILE then
                TEXTLINE := LASTLINE
              else
                TEXTLINE := 82;
              Y := 9;
              FILLSCREEN (TEXTLINE - 6);
            end;    {#79 case}
        end;   {case CH}
      end;   {#27}
    end;   {case CH of}

* * * WORD WRAP PORTION OF THE WORDPROCESSING ROUTINE * * *
* At the end of the line, if the user is still typing,    *
* this section causes the line to wrap around to the next *
* line.                                                   *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

    if (X > 50) and (TEXTLINE < 123) and (TEXTLINE > 81)
     then begin
      X := 50;
      if WORDPROC[TEXTLINE,X] <> chr(32) then begin
               {test last char in line for     }
               {blank                          }
        while WORDPROC[TEXTLINE,X] <> chr(32) do begin
          X := X - 1;  {reset X to pos of last blank  }
        end;  {while}
        for M := (X + 1) to 50 do begin
          WORDPROC[TEXTLINE+1,M-X] := WORDPROC[TEXTLINE,M];
                   {move the char to correct }
          gotoxy(M,Y);  {array pos                 }
          WORDPROC[TEXTLINE,M] := chr(32);
          write(WORDPROC[TEXTLINE,M]); {erase word from end
                                              of line}
          if Y < 9 then begin
            gotoxy(M-X,Y+1);
            write(WORDPROC[TEXTLINE+1,M-X]);
                 {write word at front of new line}
          end; {if}
        end;    {for}
        X := (M-X) +1;
      end   {if}
      else
        X := 1;
      TEXTLINE := TEXTLINE + 1;
      if Y < 9 then
        Y := Y + 1
      else begin
        Y := 4;
        FILLSCREEN (TEXTLINE - 1);
        sound(2000);delay(50);nosound;
        sound(800);delay(50);nosound;
        sound(1200);delay(50);nosound;
        sound(2000);delay(50);nosound;
      end;  {else}
    end  {if}
  until ENDRUN;

* * * SAVE FILE PORTION * * * * * * * * * * * * * * * * * * * *
* At the end of the wordprocessing session, the file is        *
* saved by moving all text lines to the end of the file        *
* so they can be readjusted when the file is next opened.      *
* A date/time/signature line is added at the end of each       *
* session to identify it.                                      *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

  if USEDFILE or NEWFILE then begin
    clrscr;
    textcolor(15); textbackground(0);
    gotoxy(13,5);
    write(' SAVING CHATTERBOX FILE ');
    if USEDFILE then begin
      LASTLINE := LASTLINE + 2;              {make room for
                                        date/time line   }
      for I := 1 to 3 do                     {save user name in
                                        hidden file  }
        WORDPROC[(LASTLINE),(I+50)] := copy(USERNAME,I,1);
      if ANONIMITY = 'A' then
        TEMPLINE := WRITEDATE('****')
      else
        TEMPLINE := WRITEDATE(USERNAME); {get date/time line
                                                for file}
      for J := 2 to 49 do
        DATELINE[J] := copy(TEMPLINE,J,1);
      for J := 2 to 49 do
        WORDPROC [(LASTLINE),J] := (DATELINE[J]);
    end;  {if USEDFILE and not NEWFILE}

    SAVEFILE :=
     concat(FILEDRIVE,':',PROBFILE,ALTERNATIVE,'.zzz');
    assign (TEXTFILE,SAVEFILE);          {open SAVEFILE and
    if NEWFILE then
      rewrite (TEXTFILE)
    else begin
      {$I-}
      append (TEXTFILE);
      {$I+}
      if IOresult <> 0 then
        rewrite(TEXTFILE);
    end;  {else}
    for J := 81 to LASTLINE do begin
      for K := 1 to 55 do
        TEMPARRAY[K] := WORDPROC[J,K];
      SAVELINE[J-80] := TEMPARRAY;
    end;  {for J := 81 to LASTLINE}
    for J := 81 to LASTLINE do begin
      writeln (TEXTFILE,SAVELINE[J-80]);
    end;  {for J}
    close(TEXTFILE);

* * * USER FILE UPDATE * * * * * * * * * * * * * * * * * *
* This section updates the file containing the name of    *
* the last user of the chatterbox.                        *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

    CHATRFILE :=
     concat(FILEDRIVE,':',PROBFILE,ALTERNATIVE,'.zzq');
    assign(CHECKFILE,CHATRFILE);
    rewrite(CHECKFILE);
    write(CHECKFILE,PERSNAME);
    close(CHECKFILE);
  end;  {if USEDFILE}

* * * IN-USE FLAG UPDATE * * * * * * * * * * * * * * * * *
* This section updates the file indicating that the       *
* chatterbox file is available for use.                   *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

  if OKAY_TO_CHAT then begin
    CHATRFILE :=
     concat(FILEDRIVE,':',PROBFILE,ALTERNATIVE,'.zzw');
    assign(CHECKFILE,CHATRFILE);
    USERFILE := concat('C',ANONIMITY);
    rewrite(CHECKFILE);
    write(CHECKFILE,USERFILE);
    close(CHECKFILE);
  end;  {if OKAY_TO_CHAT}
```

```
* * * RETURN TO NORMAL ROUTINE * * * * * * * * * * * * * *
* Clears the chatterbox window and rewrites the screen    *
* portion that was saved when chatterbox was invoked.     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

  window (23,12,78,22);
  clrscr;
  window (1,1,80,25);
  WRITESCREEN (23,12);  {write previous screen back}
end;    {procedure CHATRBOX}
```

```
{FILTER6.LIB}

procedure NewNumber(var Names  :  CritArray;  Limmit :
Integer);

(************************************************************
 *  PROCEDURE           :  NEWNUMBER                        *
 *  SUPPORTS PROGRAM    :  CTOUCH.PAS                       *
 *  LOCAL VARIABLES     :  FLAG1RENUMBER, FLAG2RENUMBER,    *
 *                         FLAG3RENUMBER                    *
 *  GLOBAL VARIABLES    :  TRACK1, LIMMIT, NAMES, L, M, N,  *
 *                         PROBLEMFLAG                      *
 *  ARRAYS USED         :  CRITARRAY                        *
 *  FILES ACCESSED      :  KRITERIAFILE                     *
 *  EXTERNAL CALLS      :  NONE                             *
 *  EXTERNAL FILTERS    :  NONE                             *
 *  CALLED FROM         :  LOADARRAY, NEWWRITE, CHANGERECORD *
 *  PURPOSE             :  RENUMBERS EVERYTHING IN THE EVENT *
 *                         THAT A RECORD HAS BEEN DELETED OR *
 *                         IF THE USER IS AT THE POINT WHERE *
 *                         HIS FILE HAS BEEN MERGED  WITH    *
 *                         OTHER COMMITTEE MEMBERS          *
 ***********************************************************)

var
   FLAG1RENUMBER, FLAG2RENUMBER, FLAG3RENUMBER  :  INTEGER;

   begin   {NewNumber}

      reset(kriteriafile);
      if filesize(kriteriafile) > 0 then
         begin    {if filesize}

            Track1 := 1;   Flag1ReNumber := 0;

            repeat

               Names[Track1].StatFlag := problemFlag;

               case names[Track1].flag1 of

                  1..100  :  begin   {Inside case flag1}
                                 if names[Track1].flag2 = 0
                                    then
                                    begin
                             {Renumbering of Major Criteria}
                                       Flag1ReNumber :=
                                         Flag1ReNumber + 1;
                                       names[Track1].flag1 :=
                                         Flag1ReNumber;
                                       Flag2ReNumber := 0;
                                       Flag3ReNumber := 0;
```

```
                                   end;     {Renumbering of
                                             Major Criteria}

                case names[Track1].flag2 of

                   1..100  :  begin    {Inside case flag2}
                                 if names[Track1].flag3 = 0
                                   then
                                    begin    {Renumbering of
                                              Sub Criteria}
                                       Flag2ReNumber :=
                                        Flag2ReNumber + 1;
                                       names[Track1].flag1 :=
                                        Flag1ReNumber;
                                       names[Track1].flag2 :=
                                        Flag2ReNumber;
                                    end;     {Renumbering of
                                              Sub Criteria}

                case names[Track1].flag3 of

                   1..100  :  begin    {Inside case flag3}
                                 Flag3ReNumber :=
                                   Flag3ReNumber + 1;
                                 names[Track1].flag1 :=
                                   Flag1ReNumber;
                                 names[Track1].flag2 :=
                                   Flag2ReNumber;
                                 names[Track1].flag3 :=
                                   Flag3ReNumber;
                              end;     {Inside case flag3}

                end;     {case statement flag3}

                end;     {Inside case flag2}

                end;     {case statement flag2}

                end;     {Inside case flag1}

                end;     {case statement flag1}

                l := Names[Track1].Flag1 * 100;
                m := Names[Track1].Flag2 * 10;
                n := Names[Track1].Flag3;
                Names[Track1].CheckPoint := l + m + n;
                Track1 := Track1 + 1;

             until Track1 = Limmit;

          end;     {if filesize}

          close(kriteriafile);

    end;    {NewNumber}
```

```
procedure Switch(Var STU1, STU2  : CriRec);

   var
      TEMPSTU : Crirec;

   begin   {Switch}
      Tempstu := Stu1;      Stu1 := Stu2;
         Stu2 := Tempstu;
   end;    {Switch}


procedure CritSort(var Names : CritArray;  limmit :
integer);

(************************************************************
 *  PROCEDURE          :  CRITSORT                        *
 *  SUPPORTS PROGRAM   :  CTOUCH.PAS, FLAGSET.PAS         *
 *  LOCAL VARIABLES    :  NOEXCHANGES, FURST, PASS, LIMID *
 *  GLOBAL VARIABLES   :  LIMMIT, NAMES                   *
 *  ARRAYS USED        :  CRITARRAY                       *
 *  FILES ACCESSED     :  NONE                            *
 *  EXTERNAL CALLS     :  SWITCH                          *
 *  EXTERNAL FILTERS   :  NONE                            *
 *  CALLED FROM        :  ALLTOGETHER, LOADARRAY, NEWWRITE *
 *  PURPOSE            :  THIS PROCEDURE DOES A NUMERIC SORT*
 *                        THAT KEEPS ALL MAJOR CRITERIA AND *
 *                        SUBSETS OF EACH MAJOR CRITERIA   *
 *                        TOGETHER.  THE SORT IS MADE ON THE*
 *                        CHECKPOINT PORTION OF THE RECORD *
 *                         'CRIREC'.                       *
 ************************************************************)

   var
      NOEXCHANGES           :  BOOLEAN;
      FURST, PASS, LIMID  :  INTEGER;

   begin   {CritSort}

      limid := limmit - 1;  Pass := 1;
      if limid > 1 then
         begin
            repeat

               Noexchanges := True;

               for Furst := 1 to limid - Pass do

               if (Names[Furst].checkpoint > Names[Furst +
                1].checkpoint) then

                  begin   {Exchange}
                     Switch(Names[Furst], Names[Furst + 1]);
                     Noexchanges := False;
                  end;    {Exchange}

                Pass := Pass + 1;

              until Noexchanges;
            end;
        end;    {CritSort}


    procedure BubbleSort(var Names : CritArray;
                             Limmit : integer);

    (*************************************************************
     *  PROCEDURE          :  BUBBLESORT                         *
     *  SUPPORTS PROGRAM   :  CTOUCH.PAS, FLAGSET.PAS            *
     *  LOCAL VARIABLES    :  NOEXCHANGES, FURST, PASS, LIMID    *
     *  GLOBAL VARIABLES   :  LIMMIT, NAMES                      *
     *  ARRAYS USED        :  CRITARRAY                          *
     *  FILES ACCESSED     :  NONE                               *
     *  EXTERNAL CALLS     :  SWITCH                             *
     *  EXTERNAL FILTERS   :  NONE                               *
     *  CALLED FROM        :  ALLTOGETHER                        *
     *  PURPOSE            :  THIS PROCEDURE DOES AN ALPHA SORT  *
     *                        THAT FLAGS ALL DUPLICATE MAJOR     *
     *                        CRITERIA BY PLACING A Ø IN THE     *
     *                        FLAG1 PORTION FO THE RECORD        *
     *                        'CRIREC'                           *
     *************************************************************)


        var
          NOEXCHANGES          :  BOOLEAN;
          FURST, PASS, LIMID   :  INTEGER;

        begin   {BubbleSort}

          limid := limmit - 1;  Pass := 1;

          if limid > 1 then
            begin
          repeat

            Noexchanges := True;
            for Furst := 1 to Limid - Pass do
            if (Names[Furst].Critname >
                 Names[Furst + 1].critname) then
               begin   {Exchange}
                 Switch(Names[Furst], Names[Furst + 1]);
                 Noexchanges := False;
               end;    {Exchange}

            Pass := Pass + 1;

          until Noexchanges;
```

```
      end;
    end;    {BubbleSort}

  procedure Odometer;


 (**********************************************************
  *  PROCEDURE           :  ODOMETER                        *
  *  SUPPORTS PROGRAM    :  CTOUCH                          *
  *  LOCAL VARIABLES     :  NONE                            *
  *  GLOBAL VARIABLES    :  ALTERNATIVE, PROBLEMFLAG        *
  *  ARRAYS USED         :  NONE                            *
  *  FILES ACCESSED      :  NONE                            *
  *  EXTERNAL CALLS      :  NONE                            *
  *  EXTERNAL FILTERS    :  NONE                            *
  *  CALLED FROM         :  LOADARRAY, INITVARIABLES, WINDOW3 *
  *  PURPOSE             :  THIS PROCEDURE WRITES TO THE    *
  *                         BOTTOM OF THE SCREEN TELLING THE *
  *                         USER AT WHAT STAGE HE IS IN.    *
  **********************************************************)


    begin   {Odometer}
      pt1 := 1;  pt2 := 1;  pt3 := 78;  pt4 := 25;
      window(pt1,pt2,pt3,pt4);
      textbackground(red);

      case ProblemFlag of

        'a' :  if alternative = 'A' then
                 begin
                   gotoXY(16,24);  write(' Input ');
                 end
               else
                 begin
                   gotoXY(2,24);  write(' Major ');
                 end;

        'b' :  if alternative = 'C' then
                 begin
                   gotoXY(9,24);
                   write(' Sub Criteria ');
                 end;

        'c' :  if alternative = 'C' then
                 begin
                   gotoXY(23,24);
                   write(' Tertiary Criteria ');
                 end;

        'h' :  if alternative = 'A' then
                 begin
                   gotoXY(16,24);  write(' Input ');
                   gotoXY(32,24);  write(' Holding ');
                 end
```

```
          else
             begin
                gotoXY(2,24);   write(' Major ');
                gotoXY(49,24);  write(' Holding ');
             end;

'i' :     if alternative = 'A' then
             begin
                gotoXY(16,24);  write(' Input ');
                gotoXY(42,24);
                write(' Review Alternatives ');
             end
          else
             begin
                gotoXY(2,24);   write(' Major ');
                gotoXY(58,24);
                write(' Review Criteria ');
             end;

'j' :     if alternative = 'A' then
             begin
                gotoXY(16,24);  write(' Input ');
                gotoXY(24,24);  write(' Final ');
             end
          else
             begin
                gotoXY(2,24);   write(' Major ');
                gotoXY(42,24);  write(' Final ');
             end;

'k' :     if alternative = 'C' then
             begin
                gotoXY(9,24);
                write(' Sub Criteria ');
                gotoXY(49,24);  write(' Holding ');
             end;

'l' :     if alternative = 'C' then
             begin
                gotoXY(9,24);
                write(' Sub Criteria ');
                gotoXY(58,24);
                write(' Review Criteria ');
             end;

'm' :     if alternative = 'C' then
             begin
                gotoXY(9,24);
                write(' Sub Criteria ');
                gotoXY(42,24);  write(' Final ');
             end;

'n' :     if alternative = 'C' then
             begin
                gotoXY(23,24);
```

```
                    write(' Tertiary Criteria ');
                    gotoXY(49,24);  write(' Holding ');
                 end;

      'o' :   if alternative = 'C' then
                 begin
                    gotoXY(23,24);
                    write(' Tertiary Criteria ');
                    gotoXY(58,24);
                    write(' Review Criteria ');
                 end;

      'p' :   if alternative = 'C' then
                 begin
                    gotoXY(23,24);
                    write(' Tertiary Criteria ');
                    gotoXY(42,24);  write(' Final ');
                 end;

      'z' :   if alternative = 'A' then
                 begin
                    textbackground(blue);
                    gotoXY(2,24);  clreol;
                    gotoXY(31,24);
                    textbackground(red);
                    write(' Input Completed ');
                 end
              else
                 begin
                    gotoXY(42,24);  write(' Final ');
                    gotoXY(58,24);
                    write(' Completed       ');
                 end;

   end;    {Case Statement}

   textbackground(blue);
   pt1 := 2;  pt2 := 2;  pt3 := 77;  pt4 := 21;
   window(pt1,pt2,pt3,pt4);

end;    {Odometer}
```

```
(**********************************************************)
   FILE        : FILTER7.LIB  (     )
   WRITTEN BY  : Mike Neeley & Bob Wooldridge, May,86
   PURPOSE     : Procedure library for TOUCHSTONE (COOP
                 Criteria Filter Program) written as a part
                 of a thesis for a Master of Science in
                 Computer Systems Management, Naval
                 Postgraduate School, Monterey, California
   CONTENTS    : ENCODE, DECODE, INTROSCREEN
                 CHANGESTATUS, CHANGECODE

(**********************************************************)

   PROCEDURE   : ENCODE
   WRITTEN BY  : Mike Neeley & Bob Wooldridge, May,86
   PURPOSE     : Encodes user name and user ID for filing
   PARAMETERS  : input: NAMECODE : array[1..8] of char;
   EXTERNAL
   NEEDS       : none

(**********************************************************)

function ENCODE(NAMECODE : CODEARRAY) : CODEARRAY;

  var
    TEMPCODE         : array[1..12] of char;
    I                : integer;

  begin
    for I := 1 to 12 do begin          {change input to all
                                         caps and}
      if NAMECODE[I] in ['a'..'z'] then   {delete non-
                                            letters}
        NAMECODE[I] := chr(ord(NAMECODE[I]) - 32);
      if not (NAMECODE[I] in ['A'..'Z','*']) then
        NAMECODE[I] := chr(32);
    end;  {for I}
          {encode all charters into code}
    for I := 1 to 12 do
      TEMPCODE[I] := chr(ord(NAMECODE[I]) + (97+I));

    ENCODE := TEMPCODE;
  end;   {procedure ENCODE}

(**********************************************************)
   PROCEDURE   : DECODE
   WRITTEN BY  : Mike Neeley & Bob Wooldridge, May,86
   PURPOSE     : Decodes user name and user ID from file
   PARAMETERS  : input: NAMECODE : array[1..8] of char:
   EXTERNAL
   NEEDS       : none
(**********************************************************)
```

```
function DECODE(NAMECODE : CODEARRAY) : CODEARRAY;

  var
    TEMPCODE : array[1..12] of char;
    I : integer;

begin
           {decode all charters from code}
    for I := 1 to 12 do
      TEMPCODE[I] := chr(ord(NAMECODE[I]) - (97+I));

    DECODE := TEMPCODE;
  end;   {procedure DECODE}


(*************************************************************)
   PROCEDURE   : INTROSCREEN
   WRITTEN BY  : Mike Neeley & Bob Wooldridge, May,86
   PURPOSE     : Draws the box for the various introductory
                 and menu screens
   PARAMETERS  : none
   EXTERNAL
   NEEDS       : Include file FILTER1.LIB
(*************************************************************)

procedure INTROSCREEN;

  begin  {procedure INTROSCREEN}
    textbackground(blue); textcolor(white);
    window(1,1,80,25);
    clrscr;
    BASICBOX(5,3,75,22);
    gotoxy(30,3);
    textbackground(red); textcolor(yellow);
    write ('      TOUCHSTONE      ');
    textbackground(blue); textcolor(white);
    window(12,5,73,20);
    pt1 := 12; pt2 := 5; pt3 := 73; pt4 := 20;
  end;

(*************************************************************)
   PROCEDURE   : CHANGESTATUS
   WRITTEN BY  : Mike Neeley & Bob Wooldridge, May,86
   PURPOSE     : Change the status of invocators/committee
                 members and add/delete persons; change
                 problem invocator password
   PARAMETERS  : none
   EXTERNAL
   NEEDS       : none
(*************************************************************)
```

```
procedure CHANGESTATUS;

  var
    NAME_OK, IOCHECK,
    CONTINUE, CHANGE          : boolean;
    X, COUNTER, K, L,
    LASTLINE                  : integer;
    CH                        : char;
    WORKFILE                  : text;
    NAMESTRING                : string[3];
    CHECKFILE                 : string[14];
    MASTER                    : string[17];
    CHECKNAME                 : array [1..3] of char;
    CHECKCODE                 : array [1..8] of char;
    CODEMASTER                : array[1..85] of char;
    CODENAME                  : array[1..85] of string[3];
    CODEWORD                  : array[1..85] of string[8];
    SAVELINE                  : array[1..85] of string[12];
    TEMPLINE                  : CODEARRAY;


  procedure HELPKEY;
    {puts up helpscreen if called}

    var
      X, Y                    : integer;

    begin
      read(kbd,CH);
      case CH of
        #59  :  begin   { F1 }
                  X := wherex; Y := whereY;
                  ScrollBox(12,8,HELPSIZE,HELPER);
                  window(pt1,pt2,pt3,pt4);
                  textcolor(white); textbackground(blue);
                  gotoxy(X,Y);
                end;    { F1 }
      end;  {CH}
    end;  {procedure CH}

  procedure GETANS;
    {solicits an answer from the user}

    begin
      repeat
        read(kbd,CH);
        if CH = #27 then
          HELPKEY;
        if CH in ['a'..'z'] then
          CH := chr(ord(CH)-32);
      until CH in ['A'..'Z',' ',#13];
    end;   {procedure GETANS}
```

```
procedure GETANSWER (A,B,C,D : char);
  {solicits an answer from the user}

  begin
    repeat
      read(kbd,CH);
      if CH = #27 then
        HELPKEY;
      if CH in [A,B] then
        CH := chr(ord(CH)-32);
    until CH in [C,D];
    write (CH); delay(500);
  end;    {procedure GETANSWER}

procedure CLEARLINES;
  {clears lines 14 & 15}

  begin
    gotoxy (1,14); clreol;
    gotoxy (1,15); clreol;
  end;    {procedure CLEARLINES}


begin  {procedure CHANGESTATUS}
                     {put information on screen}
  INTROSCREEN;
  if SELECTED = 1 then begin
    gotoxy(13,2);
    write ('INVOCATOR MASTER CODEWORD CHANGE');
    CONTINUE := true;
  end  {if SELECTED = 3}
  else begin
    gotoxy(14,2);
    write ('INVOCATOR MASTER STATUS CHANGE');
    gotoxy(4,4);
    write ('This section of the program will allow you to
             add,');
    gotoxy(4,5);
    write ('delete, or change the status of any person you
            wish.');
    gotoxy(4,7);
    write ('Please enter the initials of the individual
            you want');
    gotoxy(4,8);
    write ('to add/delete/change <OR> press enter to
            return.');
    gotoxy(21,10);
    write ('INITIALS:  ***');
    X := 32; COUNTER := 0; CONTINUE := false;

                {get initials of individual}
    repeat
      repeat
        gotoxy (X,10);
        read(kbd,CH);
```

```
          if CH = #27 then
            HELPKEY;
        until CH in [#13,'A'..'Z','a'..'z'];
        if CH in ['a'..'z'] then
          CH := chr(ord(CH)-32);
        write(CH);
        X := X + 1;
        COUNTER := COUNTER + 1;
        if CH in ['A'..'Z'] then
          CONTINUE := true;
        if CONTINUE then begin
          CHECKNAME[COUNTER] := CH;
          if CH = #13 then begin
            for L := COUNTER to 3 do
              CHECKNAME[L] := ' ';
            COUNTER := 3;
          end;  {if CH=#13}
        end;   {if continue}
      until (CH = chr(13)) or (COUNTER = 3);

                    {check initials for reserved}
      NAMESTRING := CHECKNAME;
      if (COUNTER = 3) and ((NAMESTRING = 'ZZQ') or
          (NAMESTRING = 'ZZV') or
        (NAMESTRING = 'ZZW') or (NAMESTRING = 'ZZX') or
        (NAMESTRING = 'ZZY') or (NAMESTRING = 'ZZZ')) then
     begin
        CONTINUE := false;
        sound(4000);delay(500);nosound;
        gotoxy(14,14);
        write('SORRY, THESE INITIALS RESERVED!');
        delay(2500);
      end;  {if COUNTER=3}
    end;  {else/if SELECTED=1}

                     {put all initials on file into}
                     {an array}
    if CONTINUE then begin
      CHECKFILE := concat(FILEDRIVE,':TOUCH.ZZV');
      {read file}
      assign(WORKFILE,CHECKFILE);         {Get file of codes}
      {$I-}
      reset (WORKFILE);
      {$I+}
      if IOresult = 0 then begin
        IOCHECK := true;
        LASTLINE := 1;
                    {Read file and assign parts of
                    file to code information}
     while (not eof (WORKFILE)) and (LASTLINE < 170) do
       begin
          readln (WORKFILE,SAVELINE[LASTLINE]);
          TEMPLINE := DECODE(SAVELINE[LASTLINE]);
          CODEMASTER[LASTLINE] := copy (TEMPLINE,1,1);
          CODENAME[LASTLINE] := copy (TEMPLINE,2,3);
      CODEWORD[LASTLINE] := copy (TEMPLINE,5,8);
      LASTLINE := LASTLINE + 1;
    end;  {while not eof}
    LASTLINE := LASTLINE - 1;
  end  {if IOresult}
  else begin
    clrscr;
    gotoxy(8,8);
    write ('SORRY!!  FILE: "TOUCH.ZZV"  IS NOT ON DISK
           ',FILEDRIVE);
    gotoxy(6,10);
    write ('PLEASE REPLACE NECESARY FILE BEFORE
           CONTINUING!!');
    sound(600);delay(300);nosound;delay (5000);
    IOCHECK := false;
  end; {else}
  close(WORKFILE);

              {if touch.zzv is on disk, cont}
  if IOCHECK then begin
    if SELECTED = 2 then begin
      NAME_OK := false;
      CHANGE := false;
      L := 1;
                {check for namestring in file}
      while not(L>LASTLINE) and not NAME_OK do begin
        if CODENAME[L] = NAMESTRING then
          NAME_OK := true
        else
          NAME_OK := false;
          {check user's initials for match}
        L := L + 1;
      end;  {while not L>LASTLINE};

         {if namestring is in file....}
      if NAME_OK then begin
        L := L - 1;
        if CODEMASTER[L] = 'M' then
          MASTER := 'PROBLEM INVOCATOR'
        else
          MASTER := 'COMMITTEE MEMBER';
        gotoxy(11,12);
        write ('"',CODENAME[L],'" IS LISTED AS A ',
               MASTER);
        repeat
          gotoxy (4,14);
          write ('Do you which to (C)hange that status
                 or (D)elete');
          gotoxy (4,15);
          write ('this person from the files?   *');
          gotoxy (34,15);
          GETANSWER ('c','d','C','D');
                 {if choice is to delete member..}
          if CH = 'D' then begin
            CLEARLINES;
      gotoxy (4,14);
      write ('"',CODENAME[L],'" is about to be
              deleted from ');
      write ('the files.  Do you ');
      gotoxy (4,15);
      write ('wish to continue?   *');
      gotoxy (24,15);
      GETANSWER ('y','n','Y','N');
    end;  {if CH='D'}
  until CH in ['C','Y',#13];
       {if choice is to delete member..}
  if CH in ['Y',#13] then begin
    gotoxy (1,12); clreol;
    CLEARLINES;
    for K := L to (LASTLINE - 1)do
      SAVELINE[K] := SAVELINE[K+1];
    LASTLINE := LASTLINE - 1;
    gotoxy (10,14);
    write ('"',CODENAME[L],'" NO LONGER HAS ACCESS
            TO ');
    write ('TOUCHSTONE.');
    CHANGE := true;
  end  {if CH}
         {if choice is not to delete member..}
  else begin
    CLEARLINES;
    gotoxy (4,14);
    write ('DO YOU WANT "',CODENAME[L],'" TO BE A
            PROBLEM');
    write (' INVOCATOR OR ');
    gotoxy (4,15);
    write ('A COMMITTEE MEMBER? (P/C)  *');
    gotoxy(31,15);
    GETANSWER ('p','c','P','C');
    if ((CH='P') and (CODEMASTER[L] = 'W')) or
       ((CH='C') and (CODEMASTER[L] = 'M')) then
         CHANGE := true;
    if CH = 'P' then begin
      CODEMASTER[L] := 'M';
      if (CODEWORD[L] = CODEWORD[1]) then
        CODEWORD[L] := '********';
    end
    else
      CODEMASTER[L] := 'W';
    if CODEMASTER[L] = 'M' then
      MASTER := 'PROBLEM INVOCATOR'
    else
      MASTER := 'COMMITTEE MEMBER';
    gotoxy (1,12); clreol;
    CLEARLINES;
    gotoxy (10,14);
    write ('"',CODENAME[L],'" IS NOW A
            ',MASTER,'.');
    TEMPLINE :=
   concat(CODEMASTER[L],CODENAME[L],CODEWORD[L]);
```

```
          SAVELINE[L] := ENCODE(TEMPLINE);
        end;
      end  {if NAME_OK}
                    {if namestring is not in file..}
      else begin
        gotoxy (10,14);
        write ('"',NAMESTRING,'" IS NOT ON FILE AT THE
               PRESENT TIME');
        gotoxy (10,15);
        write ('DO YOU WISH TO ADD "',NAMESTRING,'"?
               *');
        gotoxy (38,15);
        GETANSWER ('y','n','Y','N');
        if CH = 'Y' then begin
          CLEARLINES;
          gotoxy (2,14);
          write ('"',NAMESTRING,'" NOW HAS ACCESS TO
                 TOUCHSTONE.  DO ');
          write ('YOU WANT "',NAMESTRING,'" TO');
          gotoxy (2,15);
          write('BE A PROBLEM INVOCATOR OR COMMITTEE
                 MEMBER?');
          write('  (P/C)  *');
          gotoxy (54,15);
          GETANSWER ('p','c','P','C');
          LASTLINE := LASTLINE + 1;
          L := LASTLINE;
          CODENAME[L] := NAMESTRING;
          CODEWORD[L] := '        ';
          if CH = 'P' then
            CODEMASTER[L] := 'M'
          else
            CODEMASTER[L] := 'W';
          TEMPLINE :=
          concat(CODEMASTER[L],CODENAME[L],CODEWORD[L]);
          SAVELINE[L] := ENCODE(TEMPLINE);
          CHANGE := true;
        end;  {if CH = 'Y'}
      end; {else/if NAME_OK}
                    {write new file to disk}
    end {if SELECTED = 2}
    else begin
      gotoxy(4,4);
      write ('This section of the program will allow you
             to change');
      gotoxy(4,5);
      write ('the Problem Invocator Password.  Don''t
              forget that');
      gotoxy(4,6);
      write ('you will need to inform all other problem
             invocators');
      gotoxy(4,7);
      write ('of the new Password if you want them to
             have access');
      gotoxy(4,8);
```

```
write ('to Touchstone.');
gotoxy (4,10);
write ('For this version of TOUCHSTONE, that
        password is:');
gotoxy (19,11);
write ('***              ***');
gotoxy (23,11); textcolor(yellow);
textbackground(red);
write ('  ',CODEWORD[1],'  ');
textcolor(white); textbackground(blue);
gotoxy (4,12);
write ('Please input the new Problem Invocator
         password below:');
gotoxy (25,13);
write ('********');
gotoxy(16,14);
write('(Maximum of 8 letters)');
X := 25; COUNTER :=1;
{get user's codeword}
repeat    {until COUNTER >8}
  gotoxy(X,13);
  GETANS;
  CHECKCODE[COUNTER] := CH;
  if not(CHECKCODE[1] in [' ',#13]) then begin
    X := X + 1;
    write (CH);
    if CH = #13 then begin
      for L := COUNTER to 8 do
        CHECKCODE[L] := ' ';
      COUNTER := 8;
    end;  {if CH=#13}
    COUNTER := COUNTER + 1;
  end; {if not checkcode}
until COUNTER > 8;
CODEWORD[1] := CHECKCODE;

{if Problem Invocator password is the same as the
 Committee Member password, clear the Committee
 Member password}
L := 2;
while not (L>LASTLINE) do begin
  if (CODEWORD[L] = CODEWORD[1]) and
      (CODEMASTER[L] = 'M') then
       CODEWORD[L] := '********';
  L := L + 1;
end;  {while not L>LASTLINE};

gotoxy (8,15);
write ('NEW PROBLEM INVOCATOR PASSWORD IS:
        ',CODEWORD[1]);
for K := 1 to LASTLINE do begin
  TEMPLINE :=
  concat(CODEMASTER[K],CODENAME[K],CODEWORD[K]);
  SAVELINE[K] := ENCODE(TEMPLINE);
end;  {for J}
```

```
            CHANGE := true;
          end;  {else/if SELECTED=2}
          if CHANGE then begin
            assign(WORKFILE,CHECKFILE);
            rewrite (WORKFILE);
            for K := 1 to LASTLINE do begin
              writeln(WORKFILE,SAVELINE[K]);
            end;  {for J}
            close(WORKFILE);
          end;  {if CHANGE}
          delay(2000);
        end;  {if IOCHECK}
      end;  {if CONTINUE}
      clrscr;
    end;    {procedure CHANGESTATUS}
```

```
{FILTER9.LIB}

(*************************************************************
 *  PROCEDURE          :  REVIEW1                           *
 *  SUPPORTS PROGRAM   :  CTOUCH.PAS                        *
 *  LOCAL VARIABLES    :  NONE                              *
 *  GLOBAL VARIABLES   :  NAMES, LIMMIT, CH, COUNT, Y,      *
 *                        TRACK1, NUM, SECNUM, THRNUM       *
 *  ARRAYS USED        :  CRITARRAY                         *
 *  FILES ACCESSED     :  NONE                              *
 *  EXTERNAL CALLS     :  NONE                              *
 *  EXTERNAL FILTERS   :  NONE                              *
 *  CALLED FROM        :                                    *
 *  PURPOSE            :  LISTS ALTERNATIVES/CRITERIA       *
 *                        PREVIOUSLY INPUT, SO THAT THE     *
 *                        USER MAY VIEW AND OR CHANGE THE   *
 *                        RECORDS, DEPENDING UPON WHICH     *
 *                        STAGE HE IS IN.                   *
 *************************************************************)


procedure Review1(var Names  :  CritArray;
                  Limmit : Integer);

   begin   {Review}

      ch := #32;      count := 1;      Y := 6;

      gotoxy(2,6);

      repeat

         case Names[Track1].flag1 of

            1..100 :  begin   {inside case statement flag1}

                         if (Names[Track1].flag2 = 0) and
                            (Names[Track1].Flag3 = 0) then

                           begin   {Case If Statement}
                              num := names[track1].flag1;
                              gotoxy(1,Y);  clreol;
                              Write(Num,'.  ');
                              Secnum := 1;    Y := succ(Y);
                           end;   {Case If Statement}

            case Names[Track1].flag2 of

               1..100  :  begin
                  {inside case statement flag2}

                          if (Names[Track1].flag3 = 0) then
                         begin  {Case If Statement}
                            gotoxy(1,Y);  clreol;
                            gotoxy(3,Y);
                            Write(SecNum,'.   ');
                            SecNum := Succ(SecNum);
                            ThrNum := 1;    Y := succ(Y);
                         end;    {Case If Statement}

          case Names[Track1].flag3 of

             1..100  :  begin  {Case If Statement}
                           gotoxy(1,Y);  clreol;
                           gotoxy(5,Y);
                           Write(ThrNum,').   ');
                           ThrNum := ThrNum + 1;
                           Y := succ(Y);
                        end;    {Case If Statement}

          end;  {Case Statement for flag3}

               end;     {inside case statement flag2}

          end;  {Case Statement for flag2}

                   Write(Names[Track1].CritName,':   ',
                         Names[Track1].CritDef);

               end;     {inside case statement flag1}

          end;  {Case Statement for flag1}

       count := count + 1;

       Track1 := Track1 + 1;

    until (Track1 = Limmit) or (count = 14);

end;    {Review1}
```

```
procedure GetTheKeys (var Inputstring:Stringarray;
                          G:Integer);

(************************************************************
 *  PROCEDURE           :  GETTHEKEYS                        *
 *  SUPPORTS PROGRAM    :  BTOUCH.PAS, CTOUCH.PAS            *
 *  LOCAL VARIABLES     :  HORIZONTAL, VERTICAL, VERTZ       *
 *  GLOBAL VARIABLES    :  X, STOPPROG, COUNTED, HELPSIZE,   *
 *                         HELPER, COUNTER CHATOK, FILEDRIVE,*
 *                         PROBNAME, NAMESTRING, PT1, PT2,   *
 *                         PT3, PT4, INVOCATOR, CHT, TRACK1, *
 *                         SCROLLIT LIMMIT, Y, Z, G,         *
 *                         INPUTSTRING                       *
 *  ARRAYS USED         :  NONE                              *
 *  FILES ACCESSED      :  NONE                              *
 *  EXTERNAL CALLS      :  SCROLLBOX, CHATRBOX, CHATRCHECK,  *
 *                         REVIEW1                           *
 *  EXTERNAL FILTERS    :                                    *
 *  CALLED FROM         :                                    *
 *  PURPOSE             :  THIS PROCEDURE READS EACH         *
 *                         KEYSTROKE, THEREBYE REPLACING ALL *
 *                         READLNS THIS ALLOWS THE FUNCTION  *
 *                         KEYS TO BE ACCESSED AT ANY TIME   *
 *                         DURING THE PROGRAM.               *
 ************************************************************)

var
   HORIZONTAL, VERTICAL, VERTZ  :  INTEGER;

   begin    {GetTheKeys}

      StopProg := False;
      Horizontal := whereX;  Vertical := whereY;
      X := Horizontal;

      repeat

         textbackground(Yellow);
         gotoXY(X,Vertical);  write(' ');
         X := succ(X);

      until X = Horizontal + G;

      counted := 0;
      gotoXY(Horizontal,Vertical);

      for X := 1 to G do          {initialize the array}
      inputstring[X] := ' ';

      repeat

         read(kbd,cht);

         case cht of
```

```
#27  :  begin
  {Escape sequence for function keys}

             read(kbd,cht);

             case cht of

                #59  :  begin   { F1 }
                 ScrollBox(12,8,HELPSIZE,HELPER);
                 textbackground(Yellow);
                 gotoXY(Horizontal,Vertical);
                 for counter := 1 to counted do
                 write(inputstring[counter]);
                 end;    { F1 }

                #60  :  if ChatOK and
                        (Invocator <> 'M') then
                begin   { F2 }
                ChatRBox(FileDrive,ProbName,
                NameString);
                chatrcheck;
                window(pt1,pt2,pt3,pt4);
                textbackground(Yellow);
                gotoXY(Horizontal,Vertical);
                for counter := 1 to counted do
                write(inputstring [counter]);
                end;    { F2 }

                #61  :  if WeedDef and
                        (Invocator <> 'M') then
                begin   { F3 }
                ScrollBox(12,11,50,'A');
                window(pt1,pt2,pt3,pt4);
                textbackground(Yellow);
                gotoXY(Horizontal,Vertical);
                for counter := 1 to counted do
                write(inputstring [counter]);
                end;    { F3 }

                #68  :  begin   { F10 }
                StopProg := True;
                cht := #13;
                end;    { F10 }

                #71  :  if scrollit then
                begin    {home}
                textbackground(blue);
                gotoxy(2,6);  Y := 6;
                track1 := 1;
                review1(names,limmit);
                track1 := 1;
                gotoxy(2,6);   Y := 6;
```

```
if (wherey = 6) or
   (track1 = 1) then
   begin
   sound(5000);
   delay(100);
   nosound;
   end;
end;     {home}


#72  :  if scrollit then
begin    {up arrow}
textbackground(blue);
if (wherey > 6) then
begin
y := y - 1;
track1 := track1 - 1;
gotoxy(2,y);
end;
if (wherey = 6) and
(track1 > 1) then
begin
if track1 > 13 then
track1 := track1 - 13
else
track1 := track1 - 1;
review1(Names,limmit);
gotoxy(2,6);    Y := 6;
if track1 > 13 then
track1 := track1 - 13
else
track1 := 1;
end;

if (wherey = 6) and
(track1 = 1) then
begin
sound(5000);
delay(100);
nosound;
end;

end;      {up arrow}


#73  :  if scrollit then
begin    {page up}
textbackground(blue);
gotoxy(2,6);  Y := 6;
if track1 > 13 then
track1 := track1 - 17
else
track1 := 1;
if track1 < 1 then
track1 := 1;
```

```
review1(names,limmit);
if track1 > 13 then
track1 := track1 - 17
else
track1 := 1;
if track1 < 1 then
track1 := 1;
gotoxy(2,6);   Y := 6;
if (wherey = 6) or
(track1 = 1) then
begin
sound(5000);
delay(100);
nosound;
end;
end;    {page up}


#79  :  if scrollit then

        begin    {end}
           gotoxy(2,6);  Y := 6;
           textbackground(blue);
           track1 := limmit - 13;
           review1(names,limmit);
           Y := 18;
           track1 := limmit;
           gotoxy(2,18);
           if (wherey = 18) or
              (track1 = limmit)
              then
              begin
                 sound(5000);
                 delay(100);
                 nosound;
              end;
        end;    {end}


#80  :  if scrollit then

        begin    {down arrow}

           textbackground(blue);
           if (wherey < 18) and
              (wherey > 5) and
              (track1 < limmit)
               then
               begin
                  y := y + 1;
                  track1 :=
                   track1 + 1;
                  gotoxy(2,y);
               end;
```

```
          if (wherey = 18) and
             (track1 < limmit)
            then
              begin
                 if track1 > 13
                  then
                    track1 :=
                    track1 - 12
                 else
                    track1 := 1;
                 Gotoxy(2,6);
                 Y := 6;
                 review1(names,
                       limmit);
                 y := wherey;
                 gotoxy(2,y);
              end;

          if (wherey = 18) and
             (track1 = limmit)
              then
              begin
                 sound(5000);
                 delay(100);
                 nosound;
              end;

        end;      {down arrow}


#81  :  if scrollit then

        begin    {page down}

          textbackground(blue);
          gotoxy(2,6);  Y := 6;

          if track1 > 13 then
             track1 :=
                 track1 + 17;

          if track1 > limmit-13
             then
             track1 := limmit-13;

          review1(names,limmit);
          y := wherey;
          if track1 = limmit then
             Y := wherey;
          gotoxy(2,Y);
```

```
                              if (wherey = 18) or
                                 (track1 = limmit)
                                 then
                                 begin
                                    sound(5000);
                                    delay(100);
                                    nosound;
                                 end;

                           end;     {page down}


              #75, #83   :   if counted > 0 then

                              begin
                           {delete & left arrow}

                                 counted :=
                                     counted - 1;
                                 X := whereX;
                                 Z := whereY;
                                 GotoXY(X-1,Z);
                                 inputstring
                                 [counted+1] := #32;
                                 write(' ');
                                 GotoXY(X-1,Z);

                              end
                           else
                              begin
                                 gotoxy((horizontal+
                                 counted),vertical);
                                 sound(5000);
                                 delay(100);
                                 nosound;
                              end;


                end;     {case Statement}

             end;{Escape sequence for function keys}


       #32..#125   :   if counted < 6 then
                          begin   {normal characters}
                             counted := counted + 1;

                            (**************************
                             FORCES EVERY CHARACTER INTO
                             CAPS
                             *************************)

                             if cht in['a'..'z'] then
                                cht := chr(ord(cht)-32);
```

```
                                        inputstring[counted] := cht;

                                           write(cht);

                                     end;      {normal characters}

           #8          :  if (counted > 0) then

                                     begin     {backspace}

                                        counted := counted - 1;
                                        X := whereX;
                                        Z := whereY;
                                        GotoXY(X-1,Z);
                                        inputstring
                                        [counted+1] := #32;
                                        write(' ');
                                        GotoXY(X-1,Z);

                                     end      {backspace}
                                  else
                                     begin
                                        gotoxy((horizontal+
                                        counted),vertical);
                                        sound(5000);
                                        delay(100);
                                        nosound;
                                     end;

        end;  {case statement}

        if (counted = G) and (cht <> #13) then
                  {end of string}

           begin

              gotoxy((horizontal+counted),vertical);
              sound(5000);  delay(100);  nosound;

           end;

     until (cht = #13) ;

        if counted < G then

          begin
             X := Horizontal + counted;
             repeat
                textbackground(blue);
                gotoXY(X,Vertical);  write(' ');
                X := succ(X);
             until X = Horizontal + G;
          end;
```

```
      textcolor(white);
      textbackground(blue);

   end;      {GetTheKeys}


procedure Sortem(Var prob1, prob2  : probRec);

   var
      TEMPprob : probrec;

   begin   {SortEm}

      TempProb := prob1;       prob1 := prob2;
      prob2 := Tempprob;

   end;    {SortEm}


procedure probSort(var Probs : probArray;
                       limmit : integer);


(*************************************************************
 *  PROCEDURE          :  PROBSORT                           *
 *  SUPPORTS PROGRAM   :  BTOUCH.PAS                         *
 *  LOCAL VARIABLES    :  NOEXCHANGES, FURST, PASS, LIMID    *
 *  GLOBAL VARIABLES   :  PROBS, LIMMIT                      *
 *  ARRAYS USED        :  PROBARRAY                          *
 *  FILES ACCESSED     :  NONE                               *
 *  EXTERNAL CALLS     :  SORTEM                             *
 *  EXTERNAL FILTERS   :  NONE                               *
 *  CALLED FROM        :                                     *
 *  PURPOSE            :  EXECUTES AN ALPHA SORT ON RECORDS  *
 *                        IN PROBARRAY USING THE PROBLEM     *
 *                        NAME.                              *
 *************************************************************)

   var
      NOEXCHANGES          :  BOOLEAN;
      FURST, PASS, LIMID   :  INTEGER;

   begin   {probSort}

      limid := limmit - 1;  Pass := 1;

      repeat

         Noexchanges := True;

         for Furst := 1 to limid - Pass do
```

```
          if (Probs[Furst].problem >
               Probs[Furst + 1].problem) then

             begin    {Exchange}
                SortEm(Probs[Furst], Probs[Furst + 1]);
                Noexchanges := False;
             end;     {Exchange}

          Pass := Pass + 1;

       until Noexchanges;

    end;     {probSort}

 procedure ReWriteIt(var Probs  :  probArray;
                         Limmit :  Integer);


(************************************************************
 *  PROCEDURE           :  REWRITEIT                       *
 *  SUPPORTS PROGRAM    :  BTOUCH.PAS                      *
 *  LOCAL VARIABLES     :  NONE                            *
 *  GLOBAL VARIABLES    :  TRACK1, PROBS, LIMMIT, PROBNAME, *
 *                         ALTERNATIVE, CHANGEREC          *
 *  ARRAYS USED         :  PROBARRAY                       *
 *  FILES ACCESSED      :  ACITVEPROBLEMFILE = 'PROBS.TXT' *
 *  EXTERNAL CALLS      :  NONE                            *
 *  EXTERNAL FILTERS    :  NONE                            *
 *  CALLED FROM         :                                  *
 *  PURPOSE             :  REWRITES ACTIVEPROBLEMFILE      *
 *                         (PROBS.TXT)                     *
 ************************************************************)

    begin    {ReWriteIt}

       rewrite(activeproblemfile);
       Track1 := 1;

       repeat

          if (changerec = 'C') and
             (probs[track1].problem = probname) and
             (probs[track1].choice = alternative) then
             probs[track1].checkchange := changerec;

          if (changerec = 'N') and
             (probs[track1].problem = probname) and
             (probs[track1].member = namestring) and
             (probs[track1].choice = alternative) then
             probs[track1].checkchange := changerec;

          Write(activeproblemfile,Probs[Track1]);
          Track1 := Track1 + 1;

       until (Track1 = Limmit);
```

```
        close(activeproblemfile);

     end;     {ReWriteIt}


procedure LoadEmUp;

(*****************************************************************
 *   PROCEDURE          :  LOADEMUP                              *
 *   SUPPORTS PROGRAM   :  BTOUCH.PAS                            *
 *   LOCAL VARIABLES    :  NONE                                  *
 *   GLOBAL VARIABLES   :  Z, TRACK1, PROBS, LIMMIT              *
 *   ARRAYS USED        :  PROBARRAY                             *
 *   FILES ACCESSED     :  ACTIVEPROBLEMFILE = 'PROBS.TXT'       *
 *   EXTERNAL CALLS     :  PROBSORT, REWRITEIT                   *
 *   EXTERNAL FILTERS   :  NONE                                  *
 *   CALLED FROM        :                                        *
 *   PURPOSE            :  LOADS THE ARRAY PROBARRAY, SORTS      *
 *                         THE RECORDS, THEN PUTS THEM BACK      *
 *                         IN THE FILE.                          *
 *****************************************************************)

   begin    {LoadEmUp}

      Reset(ActiveProblemFile);
      z := (filesize(activeproblemfile));
      close(activeproblemfile);

      if  z > 0 then

         begin    {If the filesize statement}

            reset(activeproblemfile);

            Track1 := 1;

            while not EOF(activeproblemfile) do

               begin    {While Statement}

                  Read(activeproblemfile,Probs[Track1]);
                  Track1 := Track1 + 1;

               end;     {While Statement}

            Limmit := Track1;

            close(activeproblemfile);

            probSort(Probs,Limmit);
            rewriteit(probs,limmit);

         end;     {If the filesize statement}
   end;     {LoadEmUp}


procedure Loadthefiles;

(************************************************************
 *  PROCEDURE          :  LOADTHEFILES                      *
 *  SUPPORTS PROGRAM   :  CTOUCH.PAS                        *
 *  LOCAL VARIABLES    :  NONE                              *
 *  GLOBAL VARIABLES   :  FILEDRIVE, ALTERNATIVE,           *
 *                        NAMESTRING, PROBNAME              *
 *  ARRAYS USED        :  NONE                              *
 *  FILES ACCESSED     :  TEMPFLAGSET = 'FLAGSET.TXT'       *
 *                        (LOCAL ONLY)                      *
 *  EXTERNAL CALLS     :  NONE                              *
 *  EXTERNAL FILTERS   :  NONE                              *
 *  CALLED FROM        :  CTOUCH.PAS (MAIN PROGRAM)         *
 *  PURPOSE            :  LOADS THE TEMPFLAGSET FILE WITH   *
 *                        THE VARIABLES                     *
 *                        LISTED, SO THAT THE PROGRAM       *
 *                        FLAGSET.PAS WILL                  *
 *                        READ THE FILE AND KNOW WHAT TO DO.*
 ***********************************************************)

var
  TEMPFLAGSET  :  TEXT;

   begin    {loadthefiles}
         assign(tempflagset,'flagset.txt');
         rewrite(tempflagset);

         writeln(tempflagset,filedrive);
         writeln(tempflagset,alternative);
         writeln(tempflagset,namestring);
         writeln(tempflagset,probname);

         close(tempflagset);
   end;     {loadthefiles}
```

```
procedure AlternateChoice;

(****************************************************************
 *  PROCEDURE          :  ALTERNATECHOICE                       *
 *  SUPPORTS PROGRAM   :  BTOUCH.PAS, CTOUCH.PAS                *
 *  LOCAL VARIABLES    :  CHM                                   *
 *  GLOBAL VARIABLES   :  INPUTSTRING, ALTERNATIVE              *
 *  ARRAYS USED        :  NONE                                  *
 *  FILES ACCESSED     :  NONE                                  *
 *  EXTERNAL CALLS     :  GETTHEKEYS                            *
 *  EXTERNAL FILTERS   :  NONE                                  *
 *  CALLED FROM        :                                        *
 *  PURPOSE            :  ALLOWS THE USER TO SELECT WHETHER     *
 *                        HE WILL BE DEVELOPING                 *
 *                        ALTERNATIVES OR CRITERIA.             *
 ****************************************************************)

var
   chm  :  char;

   begin  {AlternateChoice}
      clrscr;
      gotoxy(1,8);
      write('Are you developing Alternatives or Criteria?
              A/C  ');
      gotoxy(58,8);

      repeat
         getthekeys(inputstring,1);
         alternative := inputstring;
         chm := alternative;
         gotoxy(58,8);
      until chm in ['A','C'];
   end;   {AlternateChoice}
```

```
procedure GETFILENAMES;

(**************************************************************
 *  PROCEDURE           :  GETFILENAMES                       *
 *  SUPPORTS PROGRAM    :  BTOUCH.PAS, CTOUCH.PAS             *
 *  LOCAL VARIABLES     :  AUTHORITY, TEMPNAME, CODENAME      *
 *  GLOBAL VARIABLES    :  HELPDRIVE, FILEDRIVE, NAMESTRING,  *
 *                         INVOCATOR, AUTHORIZED              *
 *  ARRAYS USED         :  NONE                               *
 *  FILES ACCESSED      :  TEMPFILE = 'DRIVEFIL.TMP'          *
 *                         (LOCAL ONLY)                       *
 *  EXTERNAL CALLS      :  DECODE                             *
 *  EXTERNAL FILTERS    :                                     *
 *  CALLED FROM         :                                     *
 *  PURPOSE             :  READS THE TEMPFILE WRITTEN IN A    *
 *                         PREVIOUS PROCEDURE AND RELOADS     *
 *                         THE GLOBAL VARIABLES.              *
 **************************************************************)

  var
    AUTHORITY                                   : char;
    TEMPNAME, CODENAME                          : string[12];
    TEMPFILE                                    : text;

  begin
    assign (TEMPFILE,'DRIVEFIL.TMP');
    {$I-}
    reset (TEMPFILE);
    {$I+}
    if IOresult = 0 then begin
      readln (TEMPFILE, CODENAME);
      TEMPNAME := DECODE (CODENAME);
      HELPDRIVE := copy(TEMPNAME,1,1);
      FILEDRIVE := copy(TEMPNAME,2,1);
      AUTHORITY := copy(TEMPNAME,3,1);
      NAMESTRING := copy(TEMPNAME,4,3);
      INVOCATOR := copy(TEMPNAME,7,1);
      close (TEMPFILE);
      if AUTHORITY = 'T' then begin
        AUTHORIZED := true;

        if invocator = 'M' then
           begin
        AUTHORITY := 'F';
        TEMPNAME :=
         concat(HELPDRIVE,FILEDRIVE,AUTHORITY,NAMESTRING,
          INVOCATOR);
        CODENAME := ENCODE(TEMPNAME);
        rewrite(TEMPFILE);
        write(TEMPFILE,CODENAME);
        close(tempfile);
           end;
      end
      else
        AUTHORIZED := false;
    end    {if IOresult}
    else
      AUTHORIZED := false;
  end;  {procedure GETFILENAMES}
```

```
(*************************************************************)
   FILE        : FILTERA.LIB  (192 lines)
   WRITTEN BY : Mike Neeley & Bob Wooldridge, May,86
   PURPOSE     : Procedure library for TOUCHSTONE (COOP
                 Criteria Filter Program) written as a part
                 of a thesis for a Master of Science in
                 Computer Systems Management, Naval
                 Postgraduate School, Monterey, California
   CONTENTS    : TITLE, BASICBOX
(*************************************************************)

   PROCEDURE   : BASICBOX
   WRITTEN BY : Mike Neeley & Bob Wooldridge, May,86
                 Based on a program created by Mark Hayes
   PURPOSE     : Draws a box as specified by the input
                 variables
   PARAMETERS : X1,Y1,X2,Y2 : integers (box corner
                 coordinates)
   EXTERNAL
   NEEDS       : none
(*************************************************************)

procedure BASICBOX (X1,Y1,X2,Y2:integer);

   var
     BC : array[1..1,1..4] of integer;
     M,I,J : Integer;

begin
                                               {box parameters}
  BC[1,1] := X1;     BC[1,2] := Y1;     BC[1,3] := X2;
  BC[1,4] := Y2;

  for M := 1 to 1 do begin                {draw a single box as
                                            needed}
    GotoXY(BC[M,1],BC[M,2]);
    write(chr(201));
    for J := (BC[M,1]+1) to (BC[M,3]-1) do begin
      GotoXY(J,BC[M,2]);
      write(chr(205))
    end;  {for J :=}
    GotoXY(BC[M,3],BC[M,2]);
    write(chr(187));
    for I := (BC[M,2]+1) to (BC[M,4]-1) do begin
      GotoXY(BC[M,1],I);
      write(chr(186));
      GotoXY(BC[M,3],I);
      write(chr(186))
    end;  {for I :=}
    GotoXY(BC[M,1],BC[M,4]);
    write(chr(200));
    for J := (BC[M,1]+1) to (BC[M,3]-1) do begin
      GotoXY(J,BC[M,4]);
```

```
        write(chr(205))
      end;  {for J :=}
      GotoXY(BC[M,3],BC[M,4]);
      write(chr(188))
    end; {for M :=}
  end;  {procedure BASICBOX}

  (*************************************************************)
     PROCEDURE   : TITLE
     WRITTEN BY : Mike Neeley & Bob Wooldridge, May,86
                  Based on a program created by Mark Hayes
     PURPOSE     : Draws the title screens with sound input
     PARAMETERS : none
     EXTERNAL
     NEEDS       : none
  (*************************************************************)

  procedure TITLE;

    var
      NOTE,M,I,J : Integer;

  begin
    window (1,1,80,25);
    port[$03d9] := $f and 3;
    textbackground(blue); textcolor(white);
    clrscr;
    BASICBOX(14,4,60,20);
    BASICBOX(17,5,63,21);
    BASICBOX(20,6,66,22);
    textcolor(yellow);
    gotoxy (35,8);              {begin first title screen}
    write ('TOUCHSTONE');
    gotoxy (25,10);
    write ('A Criteria Development Program');
    gotoxy (23,11);
    write ('for Group Decision Support Systems');
    gotoxy (32,13);
    write ('Michael E. Neeley');
    gotoxy (30,14);
    write ('Robert T. Wooldridge');
    gotoxy (28,16);
    write ('Naval Postgraduate School');
    gotoxy (30,17);
    write ('Monterey, California');
    gotoxy (38,18);
    write ('1986');
    NOTE := 0;
    repeat                  {noise for first title screen}
      sound (1000);
      delay (500);
      sound (2000);
      delay (500);
      NOTE := NOTE + 1;
    until NOTE = 3;
  nosound;
  delay (5000);      {begin second title screen}
  port[$03d9] := $f and 4;
  gotoxy (30,8);
  write ('ADMINISTRATIVE SCIENCE');
  gotoxy (25,10);
  write ('                              ');
  gotoxy (35,10);
  write ('DEPARTMENT');
  gotoxy (23,11);
  write ('                                  ');
  gotoxy (32,12);
  write (' Thesis Advisor   ');
  gotoxy (32,13);
  write ('                  ');
  gotoxy (29,14);
  write (' Xuan Tung Bui, Ph.D. ');
  NOTE := 0;
  repeat          {noise for second title screen}
    sound (1500);
    delay (500);
    sound (750);
    delay (500);
    NOTE := NOTE + 1;
  until NOTE = 3;
  nosound;
  delay (2000);
end;  {procedure TITLE}
```

```
(**********************************************************)
   FILE        : FILTERB.LIB   (      )
   WRITTEN BY  : Mike Neeley & Bob Wooldridge, May,86
   PURPOSE     : Procedure library for TOUCHSTONE (COOP
                 Criteria Filter Program) written as a part
                 of a thesis for a Master of Science in
                 Computer Systems Management, Naval
                 Postgraduate School, Monterey, California
   CONTENTS    : ENCODE, INTROSCREEN, INTRODUCTION,
                 MAKECODE

(**********************************************************)
   PROCEDURE   : INTROSCREEN
   WRITTEN BY  : Mike Neeley & Bob Wooldridge, May,86
   PURPOSE     : Draws the box for the various introductory
                 and menu screens
   PARAMETERS  : none
   EXTERNAL
   NEEDS       : Include file FILTER1.LIB
(**********************************************************)

procedure INTROSCREEN;

  begin  {procedure INTROSCREEN}
    port[$03d9]:= $f and 8;
    textbackground(blue); textcolor(white);
    window(1,1,80,25);
    clrscr;
    BASICBOX(5,3,75,22);
    gotoxy(30,3);
    textbackground(red); textcolor(yellow);
    write ('      TOUCHSTONE      ');
    textbackground(blue); textcolor(white);
    window(12,5,73,20);
  end;

(**********************************************************)
   PROCEDURE   : ENCODE
   WRITTEN BY  : Mike Neeley & Bob Wooldridge, May,86
   PURPOSE     : Encodes user name and user ID for filing
   PARAMETERS  : input: NAMECODE : array[1..8] of char;
   EXTERNAL
   NEEDS       : none
(**********************************************************)

function ENCODE(NAMECODE : CODEARRAY) : CODEARRAY;

  var
    TEMPCODE          : array[1..12] of char;
    I                 : integer;

  begin
    for I := 1 to 12 do begin
```

```
          {change input to all caps and}
        if NAMECODE[I] in ['a'..'z'] then
          {delete non-letters}
         NAMECODE[I] := chr(ord(NAMECODE[I]) - 32);
        if not (NAMECODE[I] in ['A'..'Z']) then
          NAMECODE[I] := chr(32);
      end;  {for I}
                {encode all charters into code}
      for I := 1 to 12 do
        TEMPCODE[I] := chr(ord(NAMECODE[I]) + (97+I));

      ENCODE := TEMPCODE;
    end;   {procedure ENCODE}

  (************************************************************)
     PROCEDURE   : DECODE
     WRITTEN BY  : Mike Neeley & Bob Wooldridge, May,86
     PURPOSE     : Decodes user name and user ID from file
     PARAMETERS  : input: NAMECODE : array[1..8] of char;
     EXTERNAL
     NEEDS       : none
  (************************************************************)

  function DECODE(NAMECODE : CODEARRAY) : CODEARRAY;

    var
      TEMPCODE : array[1..12] of char;
      I : integer;

  begin
                {decode all charters from code}
      for I := 1 to 12 do
        TEMPCODE[I] := chr(ord(NAMECODE[I]) - (97+I));

      DECODE := TEMPCODE;
    end;   {procedure DECODE}

  (************************************************************)
     PROCEDURE   : MAKECODE
     WRITTEN BY  : Mike Neeley & Bob Wooldridge, May,86
     PURPOSE     : Creates a new copy of TOUCH.ZZV
     PARAMETERS  : none
     EXTERNAL
     NEEDS       : none
  (************************************************************

    procedure MAKECODE;

    var
      L, X, COUNTER         : integer;
      CH                    : char;
      INPUTWORD             : string[8];
      CHECKFILE             : string[14];
      WORKFILE              : text;
      SAVELINE              : array[1..3] of string[12];
  CHECKCODE             : array [1..8] of char;

begin  {procedure MAKECODE}
  clrscr;
  gotoxy(4,6);
  write ('The files on drive ',FILEDRIVE,
         ' have not yet ');
  write ('been initialized.');
  gotoxy (4,7);
  write ('For these files, you will need a master
          password.  ');
  gotoxy (4,8);
  write ('Please input one now:    (Maximum of 8
         letters)');
  COUNTER := 1; X := 24;
  gotoxy (24,10); write ('********');
  repeat    {until COUNTER >8}
    gotoxy(X,10);
    repeat
      read(kbd,CH);
      if CH in ['a'..'z'] then
        CH := chr(ord(CH)-32);
    until CH in ['A'..'Z',' ',#13];
    write (CH);
    CHECKCODE[COUNTER] := CH;
    if not(CHECKCODE[1] in [' ',#13]) then begin
      X := X + 1;
      if CH = #13 then begin
        for L := COUNTER to 8 do
          CHECKCODE[L] := ' ';
        COUNTER := 8;
      end;  {if CH=#13}
      COUNTER := COUNTER + 1;
    end; {if not checkcode}
  until COUNTER > 8;
  INPUTWORD := CHECKCODE;
  CHECKFILE := concat(FILEDRIVE,':TOUCH.ZZV');
  assign (WORKFILE,CHECKFILE);
  rewrite (WORKFILE);
                {Read file and assign parts of
                file to code information}
  SAVELINE[1] := ENCODE(concat('     ',INPUTWORD));
  writeln(WORKFILE, SAVELINE[1]);
  CLOSE(WORKFILE);
  clrscr;
end;   {procedure MAKECODE}
```

```
(*************************************************************)
   PROCEDURE   : CHECKTHEFILES
   WRITTEN BY  : Mike Neeley & Bob Wooldridge, May,86
   PURPOSE     : Checks to see of necessary files are on
                 filedrive
   PARAMETERS  : none
   EXTERNAL
   NEEDS       : HELPDRIVE,FILEDRIVE : char;
(*************************************************************)

procedure CHECKTHEFILES;

  var
    WORKFILE                                     : text;
    CHECKFILE                                    : string[14];

  begin

    {see if TOUCH.ZZV is on the filedrive disk}
    CHECKFILE := concat(FILEDRIVE,':TOUCH.ZZV');

                                                 {read file}
    assign(WORKFILE,CHECKFILE);                  {Get file of codes}
    {$I-}
    reset (WORKFILE);
    {$I+}
    if IOresult <> 0 then begin
      MAKECODE;
    end;   {if IOresult <>0}
    close(WORKFILE);

    CHECKFILE := concat(FILEDRIVE,':PROBS.TXT');

                                                 {read file}
    assign(WORKFILE,CHECKFILE);                  {Get file of codes}
    {$I-}
    reset (WORKFILE);
    {$I+}
    if IOresult <> 0 then begin
      rewrite (WORKFILE);
    end;   {if IOresult <>0}
    close(WORKFILE);
  end;    {procedure CHECKTHEFILES}
```

```
(*************************************************************)
   PROCEDURE    : GETTHEDATE
   WRITTEN BY : Mike Neeley & Bob Wooldridge, May,86
   PURPOSE      : Gets date from registers, writes date to a
                  file
   PARAMETERS : none
   EXTERNAL
   NEEDS        : none
(*************************************************************)

procedure GetTheDate;
  {gets and changes the date}

  type
    REGISTERS = record
                  AX,BX,CX,DX,BP,SI,DS,ES,FLAGS : integer;
                end;    {record}
    STRING2 = string[2];
    STRING4 = string[4];

  var
    CONTINUE                      : boolean;
    I, MOT, CODE,DH,DL,X,
    MONTH,DAY                     : integer;
    HEXNUMBER, YEAR               : integer;
    CH                            : char;
    DATEFILE                      : text;
    DA,MO,HR,MN                   : STRING2;
    YR,HEXLINE                    : STRING4;
    STRDATE                       : string[10];
    DATE                          : string[12];
    NUMCHAR                       : array[1..8] of char;
    REGS                          : REGISTERS;

  function HEXCHANGE (HEXLINE:STRING4):integer;

    var
      B,C,D                                  : char;
      X,Y,Z,CODE                             : integer;

    begin
      B := copy (HEXLINE,2,1);
      C := copy (HEXLINE,3,1);
      D := copy (HEXLINE,4,1);
      case B of
        'A' : X := 10;
        'B' : X := 11;
        'C' : X := 12;
        'D' : X := 13;
        'E' : X := 14;
        'F' : X := 15;
      else
        val(B,X,CODE);
      end;  {base 8 of}
      case C of
        'A' : Y := 10;
        'B' : Y := 11;
        'C' : Y := 12;
        'D' : Y := 13;
        'E' : Y := 14;
        'F' : Y := 15;
      else
        val(C,Y,CODE);
      end;  {base C of}
      case D of
        'A' : Z := 10;
        'B' : Z := 11;
        'C' : Z := 12;
        'D' : Z := 13;
        'E' : Z := 14;
        'F' : Z := 15;
      else
        val(D,Z,CODE);
      end;  {base D of}
      HEXCHANGE := (16*16*X)+(16*Y)+Z;
    end;  {function HEXCHANGE}


  function HEX (DATENUM:integer):string2;

    var
      HEXDATE : string2;

    begin
      case DATENUM of
        1 : HEXDATE := '01';
        2 : HEXDATE := '02';
        3 : HEXDATE := '03';
        4 : HEXDATE := '04';
        5 : HEXDATE := '05';
        6 : HEXDATE := '06';
        7 : HEXDATE := '07';
        8 : HEXDATE := '08';
        9 : HEXDATE := '09';
        10 : HEXDATE := '0A';
        11 : HEXDATE := '0B';
        12 : HEXDATE := '0C';
        13 : HEXDATE := '0D';
        14 : HEXDATE := '0E';
        15 : HEXDATE := '0F';
        16 : HEXDATE := '10';
        17 : HEXDATE := '11';
        18 : HEXDATE := '12';
        19 : HEXDATE := '13';
        20 : HEXDATE := '14';
        21 : HEXDATE := '15';
        22 : HEXDATE := '16';
        23 : HEXDATE := '17';
        24 : HEXDATE := '18';
        25 : HEXDATE := '19';
        26 : HEXDATE := '1A';
        27 : HEXDATE := '1B';
        28 : HEXDATE := '1C';
        29 : HEXDATE := '1D';
        30 : HEXDATE := '1E';
        31 : HEXDATE := '1F';
      end;  {case DATENUM}
      HEX := HEXDATE;
    end;    {function HEX}


  function SPOT(X:integer):integer;

    var
      TEMPSPOT                                    : integer;

    begin
      case X of
        1 : TEMPSPOT := 23;
        2 : TEMPSPOT := 24;
        3 : TEMPSPOT := 31;
        4 : TEMPSPOT := 32;
        5 : TEMPSPOT := 40;
        6 : TEMPSPOT := 41;
        7 : TEMPSPOT := 42;
        8 : TEMPSPOT := 43;
      end;
      SPOT := TEMPSPOT;
    end;  {function SPOT}


   begin  {GetTheDate}
      with REGS do begin
        AX := $2A00;
        MSDOS(REGS);
        str(CX,YR);
        str(lo(DX),DA);
        if lo(DX) < 10 then
          DA := concat('0',DA);
        str(hi(DX),MO);
      end;  {with REGS}

      val(mo,mot,code);

      case MOT of
        01 : Date := 'Jan';
        02 : Date := 'Feb';
        03 : Date := 'Mar';
        04 : Date := 'Apr';
        05 : Date := 'May';
        06 : Date := 'Jun';
        07 : Date := 'Jul';
        08 : Date := 'Aug';
        09 : Date := 'Sep';
    10 : Date := 'Oct';
    11 : Date := 'Nov';
    12 : Date := 'Dec';
  end;    {case MOT of}

  Date := concat(Date,' ',da,', ',yr);

  assign(datefile,'date.txt');
  rewrite(datefile);
  writeln(datefile.date);
  close(datefile);

  INTROSCREEN;
  gotoxy(10,3);
  write ('THE CORRECT DATE IS VERY IMPORTANT TO THE');
  gotoxy(14,4);
  write ('PROPER FUNCTIONING OF TOUCHSTONE!');
  gotoxy(24,6);
  write (date);
  gotoxy(18,8);
  write ('Is this date correct?  Y');
  gotoxy(41,8);
  repeat
    read(kbd,CH);
    if CH in ['y','n'] then
      CH := chr(ord(CH)-32);
  until CH in ['Y','N',#13];
  write (CH);
  delay (500);

  if CH = 'N' then begin
    repeat
      continue := false;
      gotoxy(17,10);
      write ('Month **  Day **  Year ****');
      X := 1;
      repeat
        gotoxy(SPOT(X),10);
        repeat
          read(kbd,NUMCHAR[X]);
        until NUMCHAR[X] in ['0'..'9'];
        write(NUMCHAR[X]);
        X := X + 1;
      until X > 8;
      MO := concat(NUMCHAR[1].NUMCHAR[2]);
      DA := concat(NUMCHAR[3].NUMCHAR[4]);
      YR :=
      concat(NUMCHAR[5],NUMCHAR[6],
               NUMCHAR[7],NUMCHAR[8]);
      val(YR,YEAR,CODE);
      val(MO,MONTH.CODE);
      val(DA,DAY,CODE);
      if MONTH in [1 ..12] then
        CONTINUE := true;
```

```
          if (DAY in [1..31]) and CONTINUE then
            CONTINUE := true;
          if (YEAR in [1986..2020]) and CONTINUE  then
            CONTINUE := true;
          if (DAY in [31]) and (MONTH in [4,6,9,11]) then
            CONTINUE := false;
          if (MONTH in [2]) and (DAY in [29..31]) then
            CONTINUE := false;
          if (DAY in [29]) and (MONTH in [02]) and CONTINUE
          and
            (YEAR in [1988,1992,1996,2000,2000,2004,
                      2008,2012,2016,2020]) then
            CONTINUE := true;
        until CONTINUE;
        delay (500);
        clrscr;
        HEXLINE := concat(HEX(month),HEX(day));
        HEXNUMBER := HEXCHANGE(HEXLINE);
        with REGS do begin
          CX := YEAR;
          DX := HEXNUMBER;
          AX := $2B00;
          MSDOS(REGS);
        end;  {if CH = 'N'}
     end;
   end;  {getthedate}


   (***********************************************************)
   PROCEDURE   : INTRODUCTION
   WRITTEN BY : Mike Neeley & Bob Wooldridge, May,86
   PURPOSE     : Writes the introduction information on the
                 screen
   PARAMETERS : none
   EXTERNAL
   NEEDS       : HELPDRIVE,FILEDRIVE : char;
(***********************************************************)

procedure INTRODUCTION;

  var
    CH                                       : char;
    ACCURATE                                 : boolean;
    WORKFILE                                 : text;
    CHECKFILE                                : string[14];

  begin
    INTROSCREEN;
    gotoxy(1,8);
    write ('WOULD YOU LIKE AN INTRODUCTION TO TOUCHSTONE?
           (Y/N)   *');
    gotoxy(55,3);
    repeat
      read(kbd,CH);
      if CH in ['y','n'] tnen
    CH := chr(ord(CH) - 32);
until (CH in ['Y','N']);
write(CH); delay(500);
clrscr;
if CH = 'Y' then begin
  gotoxy(14,1);
  write('* INTRODUCTION & INFORMATION *');
  gotoxy (1,4);
  write ('     The TOUCHSTONE program is designed to
           assist you in');
  gotoxy (1,6);
  write ('developing functional and meaningful group
          criteria for ');
  gotoxy (1,8);
  write ('a Group Decision Support System.  Utilizing
          the TOUCHSTONE');
  gotoxy (1,10);
  write ('program, you will be able to condense a large
          list of    ');
  gotoxy (1,12);
  write ('spontaneously-considered criteria into a
          compact, well- ');
  gotoxy (1,14);
  write ('defined, GROUP-SELECTED set of criteria. ');
  gotoxy (15,16);
  write ('<PRESS ANY KEY TO CONTINUE>');
  repeat until keypressed;
  clrscr;
  gotoxy(9,1);
  write('* INTRODUCTION & INFORMATION (continued) *');
  gotoxy (1,4);
  write ('These criteria will be uniquely designed to
          assist you in');
  gotoxy (1,6);
  write ('resolving your current problem, whatever it
          might be. ');
  gotoxy (1,8);
  write ('Instructions, specific to each portion of the
          program, may ');
  gotoxy (1,10);
  write ('be called at any time by pressing the <F-1>
          ("HELP") key.');
  gotoxy (1,12);
  write ('Communication between "committee members" is
          accomplished');
  gotoxy (1,14);
  write ('via the "Chatterbox", an electronic notepad
          which is    ');
  gotoxy (15,16);
  write ('<PRESS ANY KEY TO CONTINUE>');
  repeat until keypressed;
  clrscr;
  gotoxy(9,1);
  write('* INTRODUCTION & INFORMATION (continued) *');
  gotoxy (1,4);
```

```
      write ('called by the <F-2> key.   An extended
              explanation of the ');
      gotoxy (1,6);
      write ('problem on which you are working may be seen
              by pressing ');
      gotoxy (1,8);
      write ('the <F-3> key.  Specific information for the
              use of these');
      gotoxy (1,10);
      write ('may be found on-screen at the bottom of each
              flash-up box.');
      gotoxy (4,12);
      write ('TOUCHSTONE proceeds through three levels of
              criteria ');
      gotoxy (1,14);
      write ('development.  At the end of each level, the
              individual  ');
      gotoxy (15,16);
      write ('<PRESS ANY KEY TO CONTINUE>');
      repeat until keypressed;
      clrscr;
      gotoxy(9,1);
      write('* INTRODUCTION & INFORMATION (continued) *');
      gotoxy (1,4);
      write ('criteria are combined for group decision and
              editing.  Once  ');
      gotoxy (1,6);
      write ('there is agreement on this level of criteria.
              TOUCHSTONE');
      gotoxy (1,8);
      write ('moves on to the next level and the next until
              the THIRD');
      gotoxy (1,10);
      write ('level has been completed.  Finally, there is
              an opportunity');
      gotoxy (1,12);
      write ('to edit the completed list.  This list is then
              ready for use');
      gotoxy (1,14);
      write ('with a DSS to evaluate the specifics for each
              criterion.');
      gotoxy (15,16);
      write ('<PRESS ANY KEY TO CONTINUE>');
      repeat until keypressed;
    end;  {if CH = Y}
    clrscr;
    gotoxy(18,1);
    write('* FILE INITIALIZATION *');
    gotoxy (1,4);
    write ('First, before you start, I need some vital
            information:     ');
    gotoxy (7,6);
    write ('On which drive are the HELP files located: ');
    gotoxy (5,8);
    write ('     DRIVE:  A      <Default:  Drive A> ');
    gotoxy (5,11);
    write ('On which drive are the committee
            files located: ');
    gotoxy (5,13);
    write ('     DRIVE:  B     <Default:  Drive B> ');
    ACCURATE := false;
    repeat
      gotoxy (18,8);
      repeat
        read(kbd,CH);
        if CH in ['a'..'h'] then
          CH := chr(ord(CH) - 32);
      until (CH in ['A'..'H',#13]);
      if CH = chr(13) then
        CH := 'A';
      write(CH);
      HELPDRIVE := CH;
      gotoxy (18,13);
      repeat
        read(kbd,CH);
        if CH in ['a'..'h'] then
          CH := chr(ord(CH) - 32);
        if (HELPDRIVE = 'A') and (CH = 'A') then
          CH := ' ';
      until (CH in ['A'..'H',#13]);
      if CH = chr(13) then
          CH := 'B';
      write(CH);
      FILEDRIVE := CH;
      gotoxy (8,16);
      write ('Is the above information accurate?   Y');
      gotoxy(45,16);
      repeat
        read(kbd,CH);
      until (CH in ['Y','y','N','n',#13]);
      if CH in ['y','n'] then
        CH := chr(ord(CH) - 32);
      write(CH);
      delay(200);
      if CH in ['Y','y',#13] then
        ACCURATE := true
      else begin
        gotoxy(1,16);
        clreol;
        gotoxy (18,8);  write ('A');
        gotoxy (18,13); write ('B');
      end;  {else/if CH}
    until ACCURATE;
  end;
```

```
(**********************************************************)
   FILE        : FILTERC.LIB   (     )
   WRITTEN BY  : Mike Neeley & Bob Wooldridge, May,86
   PURPOSE     : Procedure library for TOUCHSTONE (COOP
                 Criteria Filter Program) written as a part
                 of a thesis for a Master of Science in
                 Computer Systems Management, Naval
                 Postgraduate School, Monterey, California
   CONTENTS    : VERIFYCODE
(**********************************************************)

   PROCEDURE   : VERIFYCODE
   WRITTEN BY  : Mike Neeley & Bob Wooldridge, May,86
   PURPOSE     : Checks to see if user name and code are
                 valid
   PARAMETERS  : input: NAMECODE : array[1..8] of char;
   EXTERNAL
   NEEDS       : AUTHORITY : char;
(**********************************************************)

procedure VERIFYCODE;

  var
    NAME_OK, CODE_OK        : boolean;
    CONTINUE, MASTER,
    INITIALCHECK            : boolean;
    COUNTER, TRIES,
    LASTLINE                : integer;
    J, K, L, X              : integer;
    CH                      : char;
    WORKFILE                : text;
    CHECKFILE               : string[14];
    CHECKNAME               : array [1..3] of char;
    CHECKCODE               : array [1..8] of char;
    CODEMASTER              : array[1..85] of char;
    CODENAME                : array[1..85] of string[3];
    CODEWORD                : array[1..85] of string[8];
    SAVELINE                : array[1..85] of string[12];
    TEMPLINE                : CODEARRAY;

  procedure GETANSWER (A,B,C,D : char);
    {solicits an answer from the user}

    begin
      repeat
        read(kbd,CH);
        if CH in [A,B] then
          CH := chr(ord(CH)-32);
      until CH in [C,D,#13];
      write (CH);
    end;    {procedure GETANSWER}

  procedure GETANS;
```

```
    {solits an answer from the user}

    begin
      repeat
        read(kbd,CH);
        if CH in ['a'..'z'] then
          CH := chr(ord(CH)-32);
      until CH in ['A'..'Z',' ',#13];
    end;   {procedure GETANS}

    procedure CHECKANSWER(WRITECH : char);
      {gets code input}

      begin
        CHECKCODE[COUNTER] := CH;
        if not(CHECKCODE[1] in [' ',#13]) then begin
          write (WRITECH);
          X := X + 1;
          if CH = #13 then begin
            for L := COUNTER to 8 do
              CHECKCODE[L] := ' ';
            COUNTER := 8;
          end;  {if CH=#13}
          COUNTER := COUNTER + 1;
        end; {if not checkcode}
    end;  {procedure CHECKANSWER}


  procedure CHECKINITIALS (XCOORDINATE,
                           YCOORDINATE : integer);
    {checks to see if initials are valid}

    begin
      CHECKNAME[COUNTER] := CH;
      if not(CHECKNAME[1] in [' ',#13]) then begin
        write (CH);
        X := X + 1;
        if CH = #13 then begin
          for L := COUNTER to 3 do
            CHECKNAME[L] := ' ';
          COUNTER := 3;
        end;  {if CH=#13}
        COUNTER := COUNTER + 1;
      end;
      NAMECHECK := CHECKNAME;
      if (COUNTER = 4) and ((NAMECHECK = 'ZZQ') or
        (NAMECHECK = 'ZZV') or (NAMECHECK = 'ZZW') or
        (NAMECHECK = 'ZZX') or (NAMECHECK = 'ZZY') or
        ((NAMECHECK = 'ZZZ') and INITIALCHECK)) then begin
          COUNTER := 1;
          gotoxy(14,16);
          write('SORRY, THESE INITIALS RESERVED!');
          sound(4000);delay(500);nosound;
          delay(1500);
          gotoxy(14,16);
        write('                                              ');
        gotoxy(XCOORDINATE,YCOORDINATE); write('***');
        X := XCOORDINATE;
    end;  {if NAMECHECK = 'ZZZ'}
  end;  {procedure CHECKINITIALS}


begin  {procedure VERIFYCODE}
                                                  {initialize}
  X := 31;
  COUNTER := 1;
  AUTHORITY := 'F';
  CODE_OK := false;
  TRIES := 1;
  CHECKFILE := concat(FILEDRIVE,':TOUCH.ZZV');

                                                  {read file}
  assign(WORKFILE,CHECKFILE);                     {Get file of codes}
  reset (WORKFILE);
  LASTLINE := 1;
                                  {Read file and assign parts of
                                  file to code information}
  while (not eof (WORKFILE)) and (LASTLINE < 170) do begin
    readln (WORKFILE,SAVELINE[LASTLINE]);
    TEMPLINE := DECODE(SAVELINE[LASTLINE]);
    CODEMASTER[LASTLINE] := copy (TEMPLINE,1,1);
    CODENAME[LASTLINE] := copy (TEMPLINE,2,3);
    CODEWORD[LASTLINE] := copy (TEMPLINE,5,8);
    LASTLINE := LASTLINE + 1;
  end;  {while not eof}
  LASTLINE := LASTLINE - 1;
  close(WORKFILE);

  clrscr;
  if LASTLINE = 1 then begin
       {instructions to new prob. inv.}
    clrscr;
    gotoxy (13,1);
    write ('GREETINGS, NEW PROBLEM INVOCATOR!');
    gotoxy (5,3);
    write ('As the person initiating this copy of
             TOUCHSTONE,');
    gotoxy (5,4);
    write ('you are designated as the:');
    gotoxy (5,5);
    write ('                  "Problem Invocator".');
    gotoxy (5,6);
    write ('As such, you are the one to define the
            problems,');
    gotoxy (5,7);
    write ('select the committee membership, and perform
             the');
    gotoxy (5,8);
    write ('various other maintenance functions.  You may,
            of');
```

```
gotoxy (5,9);
write ('course, designate other problem invocators if
        you');
gotoxy (5,10);
write ('so desire, or maintain control by yourself.
        The');
gotoxy (5,11);
write ('choice is yours.');
gotoxy (5,13);
write ('For log-on purposes, I will need to know
        your');
gotoxy (5,14);
write ('initials (a maximum of 3):   *** ');
X := 34; INITIALCHECK := true;
repeat   {until CONTINUE}
  COUNTER := 1;
                                  {get user's initials}
  repeat
    gotoxy(X,14);
    GETANS;
    CHECKINITIALS(34,14);
    NAMESTRING := NAMECHECK;
  until COUNTER > 3;
  gotoxy (14,16);
  write ('Are these initials correct?  Y');
  gotoxy (43,16);
  GETANSWER('y','n','Y','N');
  if CH in ['Y',#13] then begin
    CONTINUE := true;
    CODENAME[2] := NAMECHECK;
  end   {if CH}
  else begin
    X := 34; gotoxy(X,14); write ('***');
    CONTINUE := false:
  end;
until CONTINUE;
clrscr;
gotoxy (3,1);
write ('Thank you for your initials.  You will need to
        use ');
gotoxy (3,2);
write ('these to identify yourself to the computer
        each time');
gotoxy (3,3);
write ('you log on.  When you do log on to TOUCHSTONE,
        you ');
gotoxy (3,4);
write ('will need to use the Problem Invocator
        Password if');
gotoxy (3,5);
write ('you wish to identify yourself as the
        problem');
gotoxy (47,5);
write (' invocator.');
gotoxy (3,6);
```

```
write ('For this version of TOUCHSTONE, that password
        is:');
gotoxy (20,7);
write ('***             ***');
gotoxy (24,7); textcolor(yellow); textbackground(red);
write ('  ',CODEWORD[1],'  ');
textcolor(white); textbackground(blue);
gotoxy (3,9);
write('(You should memorize this password for future
        use.  If');
gotoxy (3,10);
write ('you wish, you have the option to change it in
        the ');
gotoxy (3,11);
write('Problem Invocator Menu.)  If you prefer to log
        on as');
gotoxy (3,12);
write('a committee member instead, you will need a
        personal');
gotoxy (3,13);
write ('password of your own.  This word (letters
        only) can be');
gotoxy (3,14);
write ('up to 8 letters in length:   ********');
X := 32; TRIES := 0; COUNTER :=1;

            {get problem invocator's codeword}
repeat {until CONTINUE}
  CONTINUE := false;
  repeat    {until COUNTER >8}
    gotoxy(X,14);
    GETANS;
    CHECKANSWER(CH);
  until COUNTER > 8;
  gotoxy (15,16);
  write ('Is this code word correct?   Y');
  gotoxy (44,16);
  GETANSWER('y','n','Y','N');
  if CH in ['Y',#13] then
    CONTINUE := true
  else begin
    gotoxy (32,14); write ('********');
    X := 32; COUNTER := 1;
    CONTINUE := false;
  end;
until CONTINUE;
USERCODE := CHECKCODE;
CODEWORD[2] := USERCODE;
CODEMASTER[2] := 'M';
LASTLINE := 3;

                {get committee member information}
clrscr;
gotoxy(12,2);
write('**  COMMITTEE MEMBER INFORMATION  **');
```

```
gotoxy(4,4);
write ('Now is a good time to input the initials of
        those');
gotoxy(4,5);
write ('people you know will
        need to have access to ');
gotoxy(4,6);
write ('TOUCHSTONE.  Please input their initials and,
        for');
gotoxy(4,7);
write ('each, designate whether that individual is to
        be a');
gotoxy(4,8);
write ('[P]roblem invocator or merely a [C]ommittee
        member.');
gotoxy (4,9);
write ('(The default choice is Committee member.)');
gotoxy (4,11);
write ('Initials:                   Access level (P/C):
        [C]');
gotoxy (17,15);  write ('(Write `ZZZ` to exit)');
repeat   {until NAMECHECK = ZZZ}
  COUNTER := 1; NAME_OK := true;
  X := 15; gotoxy(X,11); write ('***');
  repeat   {until CONTINUE}
    {get user's initials}
    repeat   {until COUNTER >3}
      gotoxy(X,11);
      GETANS;
      INITIALCHECK := false;
      CHECKINITIALS(15,11);
      INITIALCHECK := true;
    until COUNTER > 3;
    gotoxy (14,13);
    write ('Are these initials correct?  Y');
    gotoxy (43,13);
    GETANSWER('y','n','Y','N');
    if CH in ['Y',#13] then begin
      L := 1;
      while not(L>LASTLINE) and NAME_OK do begin
        if CODENAME[L] = NAMECHECK then
          NAME_OK := false
        else
          NAME_OK := true;   {check user's initials
                                     for match}
        L := L + 1;
      end;  {while not L>LASTLINE};
      if NAME_OK then begin
        CONTINUE := true;
        CODENAME[LASTLINE] := NAMECHECK;
      end   {if NAME_OK}
      else begin
        gotoxy(14,16);
        write('SORRY, THESE INITIALS ARE USED!');
        sound(4000);delay(500);nosound;
```

```
            delay(1500);
            gotoxy(14,16);
            write('                                ');
            gotoxy(15,11); write('***');
            X := 15; COUNTER := 1;
            CONTINUE := false; NAME_OK := true;
          end;  {else}
        end   {if CH}
        else begin
          X := 15; COUNTER := 1;
          CONTINUE := false;
        end;
        gotoxy (14,13);
        write ('                                ');
      until CONTINUE;

      if NAMECHECK <> 'ZZZ' then begin
        gotoxy (52,11); write ('C');
        gotoxy(52,11);
        GETANSWER('c','p','C','P');
        if CH in ['C',#13] then
          CODEMASTER[LASTLINE] := 'W'
        else
          CODEMASTER[LASTLINE] := 'M';
        CODEWORD[LASTLINE] := '        ';
      end;   {if NAMECHECK <> 'ZZZ'}
      LASTLINE := LASTLINE + 1;
    until NAMECHECK = 'ZZZ';
    LASTLINE := LASTLINE - 1;
    assign(WORKFILE,CHECKFILE); {Rewrite file of codes}
    rewrite (WORKFILE);
    for K := 1 to LASTLINE do begin
      TEMPLINE :=
   concat(CODEMASTER[K],CODENAME[K],CODEWORD[K]);
      SAVELINE[K] := ENCODE(TEMPLINE);
      writeln(WORKFILE,SAVELINE[K]);
    end;  {for J}
    close(WORKFILE);
    AUTHORITY := 'T';
    INVOCATOR := 'M';
  end   {if LASTLINE}
  else begin         {Other than new invocator}
    X := 40;
    gotoxy(16,4);
    write('**  SIGN-ON INFORMATION  **');
    gotoxy(15,7);
    write ('What are your initials?  ***');
    repeat   {until NAME_OK or TRIES=3}
      CHECKNAME := '   ';  NAME_OK := false;
      {get user's initials}
      repeat
        gotoxy(X,7);
        GETANS;
        CHECKINITIALS(40,7);
        NAMESTRING := NAMECHECK;
      until COUNTER > 3;

      {check input name against names on file}
      J := 1;
      while not(J>LASTLINE) and not NAME_OK do begin
        if CODENAME[J] = NAMECHECK then
          NAME_OK := true;       {check user's initials
                                        for match}
        J := J + 1;
      end;  {while not J>LASTLINE};
      if not NAME_OK then begin
        COUNTER := 1;
        X := 40;
        TRIES := TRIES + 1;
        gotoxy(15,14);
        write('THESE INITIALS NOT ON FILE');
        sound(4000);delay(500);nosound;
        delay(1000);
        gotoxy(15,14);write('                                  ');
        gotoxy(40,7); write('***');
      end;   {if not NAME_OK}
      J := J - 1;
    until NAME_OK or (TRIES>3);

    {check for correct user password}
    if NAME_OK then begin
      if (CODEWORD[J] = '        ') or
         (CODEWORD[J] = '********')
        then begin
        if (CODEWORD[J] = '        ') then begin
          gotoxy(6,9);
          write('As a new TOUCHSTONE user,
                 you will need ');
          write('a password.');
          gotoxy(6,10);
          write('What would you like for your password?
                  ********');
        end  {if CODEWORD[J]}
        else begin
          gotoxy(6,9);
          write('Your Committee Member password
                 has been ');
          write('erased.  What');
          gotoxy(6,10);
          write('would you like for your new password?
                 ********');
        end;  {else/if CODEWORD[J]}
        gotoxy(19,12);
        write('(Maximum of 8 letters)');
        X := 45; TRIES := 0; COUNTER :=1;
        {get user's codeword}
        repeat {until CONTINUE}
          CONTINUE := false;
          repeat    {until COUNTER >8}
            gotoxy(X,10);
      GETANS;
      CHECKANSWER(CH);
    until COUNTER > 8;
    gotoxy (15,16);
    write ('Is this code word correct?   Y');
    gotoxy (44,16);
    GETANSWER('y','n','Y','N');
    if CH in ['Y',#13] then
      CONTINUE := true
    else begin
      gotoxy (45,10); write ('********');
      X := 45; COUNTER := 1;
      CONTINUE := false;
    end;
  until CONTINUE;
  USERCODE := CHECKCODE;
  CODEWORD[J] := USERCODE;
  TEMPLINE :=
  concat(CODEMASTER[J],CODENAME[J],CODEWORD[J]);
  SAVELINE[J] := ENCODE(TEMPLINE);
  assign(WORKFILE,CHECKFILE);
       {Get file of codes}
  rewrite (WORKFILE);
  for K := 1 to LASTLINE do begin
    writeln(WORKFILE,SAVELINE[K]);
  end;  {for J}
  close(WORKFILE);
  AUTHORITY := 'T';
  gotoxy(15,16); clreol;
  if CODEMASTER[J] = 'M' then begin
    gotoxy(12,14);
    write ('Which menu do you wish to use today?');
    gotoxy(8,15);
    write ('(P)roblem invocator or (C)ommittee
            member:  *');
    gotoxy(52,15);
    GETANSWER ('p','c','P','C');
    if CH = 'P' then begin
      gotoxy(1,9); clreol;
      gotoxy(1,10); clreol;
      gotoxy(1,14); clreol;
      gotoxy(1,15); clreol;
      gotoxy (6,10);
      write('What is your Problem Invocator
             password?    ********');
      X := 50; TRIES := 1; COUNTER := 1;
      repeat   {until CODE_OK or TRIES=3);
        {get user's codeword}
        repeat    {until COUNTER >8}
          gotoxy(X,10);
          GETANS;
          CHECKANSWER('M');
        until COUNTER > 8;
        delay(250);
```

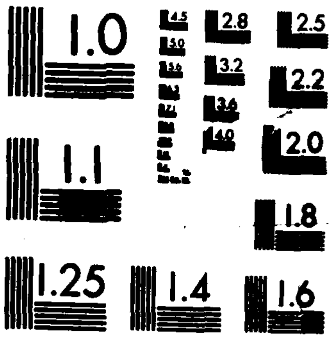
```
              {check usercode against codewords on file}
              USERCODE := CHECKCODE;
              if (CODEWORD[1] = USERCODE) then begin
                CODE_OK := true;
                INVOCATOR := 'M';
                AUTHORITY := 'T';
              end   {if MASTER}
              else begin
                AUTHORITY := 'F';
                COUNTER := 1;
                X := 50;
                sound(4000);delay(500);nosound;
                gotoxy(19,14);
                write('INCORRECT ACCESS CODE');
                delay(1000);
                gotoxy(19,14);
                write('                              ' );
                gotoxy(50,10); write('********');
                TRIES := TRIES + 1;
              end;   {else}
            until CODE_OK or (TRIES>3);
          end;  {if ch = 'P'}
          delay(500);
        end;   {if CODEMASTER[J]}
      end   {if NAME_OK}
      else begin
        if CODEMASTER[J] = 'M' then
          MASTER := true  {Person signing on is a problem}
        else                      {invocator}
          MASTER := false;
        gotoxy(6,10);
        write('What is your user (or invocator) password?
              ********');
        X := 50; TRIES := 1; COUNTER := 1;
        repeat   {until CODE_OK or TRIES=3};
          {get user's codeword}
          repeat    {until COUNTER >8}
            gotoxy(X,10);
            GETANS;
            CHECKANSWER('M');
          until COUNTER > 8;
          delay(250);

          {check usercode against codewords on file}
          USERCODE := CHECKCODE;
          if (CODEWORD[J] = USERCODE) then
            CODE_OK := true
          else
            if MASTER and (CODEWORD[1] = USERCODE) then
            begin
              CODE_OK := true;
              INVOCATOR := 'M';
            end   {if MASTER}
            else begin
              COUNTER := 1;
```

```
                    X := 50;
                    sound(4000);delay(500);nosound;
                    gotoxy(19,14);
                    write('INCORRECT ACCESS CODE');
                    delay(1000);
                    gotoxy(19,14);
                    write('                         ' );
                    gotoxy(50,10); write('********');
                    TRIES := TRIES + 1;
                  end;   {else}
              until CODE_OK or (TRIES>3);
            end;   {else}
          end;   {if NAME_OK}
          if CODE_OK then
            AUTHORITY := 'T';
        end;   {else - if LASTLINE=1}
      end;   {procedure VERIFYCODE}
```

```
{FRONTEND.LIB}


procedure NoFiles;

(*************************************************************
 *  PROCEDURE           :  NOFILES                           *
 *  SUPPORTS PROGRAM    :  BTOUCH.PAS                        *
 *  LOCAL VARIABLES     :  NONE                              *
 *  GLOBAL VARIABLES    :  STOPPROG                          *
 *  ARRAYS USED         :  NONE                              *
 *  FILES ACCESSED      :  NONE                              *
 *  EXTERNAL CALLS      :  NONE                              *
 *  EXTERNAL FILTERS    :  NONE                              *
 *  CALLED FROM         :                                    *
 *  PURPOSE             :  WRITES 'NO FILES ON DISK' ON THE  *
 *                         SCREEN AFTER THE CALLING          *
 *                         PROCEDURE CHECKS THE FILE.        *
 ************************************************************)

   begin   {nofiles}
      if not (stopprog) then
         begin    {if not stopprog}
            gotoxy(21,9);
            textbackground(red);
            write(' No Files on disk ');
            delay(4000);
            textbackground(blue);
            stopprog := true;
         end;     {if not stopprog}
   end;    {nofiles}


procedure warning;

(*************************************************************
 *  PROCEDURE           :  WARNING                           *
 *  SUPPORTS PROGRAM    :  BTOUCH.PAS                        *
 *  LOCAL VARIABLES     :  NONE                              *
 *  GLOBAL VARIABLES    :  NONE                              *
 *  ARRAYS USED         :  NONE                              *
 *  FILES ACCESSED      :  NONE                              *
 *  EXTERNAL CALLS      :  NONE                              *
 *  EXTERNAL FILTERS    :  NONE                              *
 *  CALLED FROM         :  DISPLAYIT,                        *
 *  PURPOSE             :  WRITES 'FILE NOT FOUND' AFTER     *
 *                         PROCEDURE CHECKS FILE FOR RECORD. *
 ************************************************************)

   begin   {warning}
      gotoxy(21,15);
      textbackground(red);
      write(' File not found ');
```

AD-A183 203 TOUCHSTONE: A CRITERIA DEVELOPMENT PROGRAM FOR GROUP DECISION SUPPORT SYSTEMS(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA R T WOOLDRIDGE ET AL. MAR 87 3/3

UNCLASSIFIED F/G 12/5 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```
      delay(4000);
      textbackground(blue);
      gotoxy(21,15);  clreol;
   end;    {warning}


procedure DisplayIt;

(************************************************************
 *  PROCEDURE           :  DISPLAYIT                        *
 *  SUPPORTS PROGRAM    :  BTOUCH.PAS                       *
 *  LOCAL VARIABLES     :  COUNTERS                         *
 *  GLOBAL VARIABLES    :                                   *
 *  ARRAYS USED         :  NONE                             *
 *  FILES ACCESSED      :  ACTIVEPROBLEMFILE = 'PROBS.TXT'  *
 *  EXTERNAL CALLS      :  WARNING                          *
 *  EXTERNAL FILTERS    :                                   *
 *  CALLED FROM         :                                   *
 *  PURPOSE             :  DISPLAYS SPECIFIC PROBLEM AND    *
 *                         MEMBERS ASSIGNED                 *
 ************************************************************)

var
   COUNTERS : INTEGER;

   begin   {display it}
      reset(ActiveProblemFile);
      counters := 1;            clrscr;
      while not EOF(activeproblemfile) do
         begin   {While Statement}
            read(ActiveProblemFile, members);
            if (members.problem = probname) and
               (members.choice = alternative) then
                counters := counters + 1;
         end;    {While Statement}
      close(activeproblemfile);
      reset(activeproblemfile);
      Y := 3;       moveover := 10;
      while not EOF(activeproblemfile) do
         begin   {While Statement}
            read(ActiveProblemFile, members);
            if (members.problem = probname) and
               (members.choice = alternative) then
               begin
            if counters > 10 then
               begin   {if counters > 10}
                  if (members.problem = probname) and
                     (members.choice = alternative) then
                      begin   {2nd if comparing probname}
                         gotoxy(10,1);
                         write('PROBLEM    MEMBER',
                         '       PROBLEM    MEMBER');
                         stopgap := true;
                         gotoxy(moveover,Y);
                         write(members.problem);
```

```
                              gotoxy(moveover + 12,Y);
                              write(members.member);
                              Y := Y + 1;
                              if Y = 11 then
                                 begin   {if Y > 10}
                                    Y := 3;
                                    moveover := 32;
                                 end;    {if Y > 10}
                        end;    {2nd if comparing probname}
                 end    {if counters > 10}
              else
                 begin   {if comparing probname}
                    gotoxy(20,1);
                    write('PROBLEM    MEMBER');
                    stopgap := true;
                    gotoxy(20,Y);  write(members.problem);
                    gotoxy(32,Y);  write(members.member);
                    Y := Y + 1;
                 end;    {if comparing probname}
              end;
           end;    {while statement}
        close(activeproblemfile);
        if not (stopgap) and
           not (stopprog) then
              warning;
     end;    {display it}


  procedure LoadIt;

  (************************************************************
   *  PROCEDURE          :  LOADIT                            *
   *  SUPPORTS PROGRAM   :  BTOUCH.PAS                        *
   *  LOCAL VARIABLES    :  TEMPPROB, REALLYTEMP              *
   *  GLOBAL VARIABLES   :                                    *
   *  ARRAYS USED        :  NONE                              *
   *  FILES ACCESSED     :  ACTIVEPROBLEMFILE = 'PROBS.TXT'   *
   *  EXTERNAL CALLS     :  NONE                              *
   *  EXTERNAL FILTERS   :  NONE                              *
   *  CALLED FROM        :                                    *
   *  PURPOSE            :  THIS PROCEDURE LOADS THE ACTIVE   *
   *                        PROBLEM FILE INTO AN ARRAY FOR    *
   *                        THE PURPOSE OF DELETING A MEMBER  *
   *                        FROM A SPECIFIC COMMITTEE, OR     *
   *                        DELETING A PROBLEM COMPLETELY.    *
   *                        THE PROCEDURE DISPLAYS ALL        *
   *                        RECORDS SO THAT THE USER CAN VIEW *
   *                        WHAT PROBLEMS HE MAY WISH TO      *
   *                        MANIPULATE.                       *
   ************************************************************)

  var
     tempprob    :  string8;
     reallytemp  : string8;
```

```
begin   {LoadIt}
   clrscr;          Y := 3;
   X := 1;        tempprob := ' ';
   Assign(activeproblemfile,
      concat(filedrive,':probs.txt'));
   Reset(ActiveProblemFile);
   if (filesize(activeproblemfile)) > 0 then
      begin   {If the filesize statement}
         write('PROBLEM');
         tempprob := ' ';
         while not EOF(ActiveProblemFile) do
            begin   {While statement}
               read(ActiveProblemFile,members);
               reallytemp :=
           concat(members.problem+members.choice);
               if (tempprob <> reallytemp) and
                  (members.choice = alternative) then
                  begin
                     gotoxy(X,Y);
                     write(members.problem);
                     Y := Y + 1;
                  end;
               tempprob := reallytemp;
               if Y > 10 then
                  begin
                     Y := 3;
                     X := X + 12;
                     gotoxy(x,1);  write('PROBLEM');
                  end;
            end;

         close(activeproblemfile);
      end;    {If the filesize statement}
end;    {LoadIt}
```

```
procedure DeleteAProblem;

(***********************************************************
 *  PROCEDURE           :  DELETEAPROBLEM                   *
 *  SUPPORTS PROGRAM    :  BTOUCH.PAS                       *
 *  LOCAL VARIABLES     :  SHORTNAME, TEMPMEMBER, TEMP1,    *
 *                         CHECKED, COUNTS                  *
 *  GLOBAL VARIABLES    :  Z, INPUTSTRING, STOPPROG, MEMBERS *
 *                         PROBNAME, FILEDRIVE, NEWSTRING,  *
 *                         ALTERNATIVE                      *
 *  ARRAYS USED         :  NONE                             *
 *  FILES ACCESSED      :  TEMPROBLEMFILE, CHECKFILE,       *
 *                         ACTIVEPROBLEMFILE                *
 *  EXTERNAL CALLS      :  LOADIT, WARNING, NOFILES         *
 *  EXTERNAL FILTERS    :  FILTER9.LIB                      *
 *  CALLED FROM         :  PROBMANIPULATION                 *
 *  PURPOSE             :  THIS PROCEDURE ALLOWS THE USER TO *
 *                         SELECT A PROBLEM DISPLAYED ON    *
 *                         THE SCREEN FROM THE PROCEDURE    *
 *                         'LOADIT' FOR DELETEION.  IF THE  *
 *                         USER CHANGES HIS MIND ABOUT      *
 *                         DELETING A PROBLEM, HE ONLY HAS  *
 *                         TO PUSH  THE RETURN KEY AND NO   *
 *                         FILES WILL BE DELETED.           *
 *                         CONFIRMATION OF THE DELETED      *
 *                         PROBLEM IS GIVEN AND THE         *
 *                         REMAINING PROBLEMS ARE AGAIN     *
 *                         DISPLAYED.                       *
 ***********************************************************)

var
   SHORTNAME          :  STRING[7];
   TEMPPROBLEMFILE    :  file of PROBREC;
   TEMPMEMBER         :  PROBREC;
   TEMP1              :  STRING12;
   CHECKFILE          :  TEXT;
   CHECKED            :  BOOLEAN;
   COUNTS             :  INTEGER;

   begin   {DeleteAProblem}
      checked := false;       counts := 0;       LoadIt;
      Reset(ActiveProblemFile);
      z := (filesize(activeproblemfile));
      close(activeproblemfile);
      if  z > 0 then
         begin   {If the filesize statement}
            gotoxy(1,12);
            write('CAUTION!!!  Entering a problem name from
                   this list, will');
            gotoxy(1,13);
            write('delete ALL files with that name.',
             '  To quit without deleting');
            gotoxy(1,14);      write('a problem,press F10. ');
            repeat
               gotoxy(1,16);
```

```
     write('Enter the problem
              you wish to delete:  ');
     repeat
        getthekeys(Inputstring,7);
        shortName := inputstring;
        gotoxy(40,16);
     until  (ord(shortname[1]) > 32) or (stopprog);
     a := 2;
     probname := shortName[1];
     while (shortname[a] <> chr(32)) and (a<8) do
        begin
           probname :=
          concat(probname,shortname[a]);
           a := a + 1;
        end;
     if not stopprog then
        begin    {if not stopprog}
           Assign(activeproblemfile,
   concat(filedri ve,':probs.txt'));
           reset(ActiveProblemFile);
           Assign(tempproblemfile,
         concat(filedrive ,':tempprob.txt'));
           rewrite(tempProblemFile);
           while not EOF(activeproblemfile) do
              begin   {While Statement}
                 read(ActiveProblemFile, members);
                 tempmember := members;
              if (members.problem = probname) then
                 begin
                    NewString :=
                    probname+alternative+
                    '.'+members.member;
                    Assign(kriteriafile,
                    concat(filedrive,':',
                           newstring));
                    {$I-}
                    erase(KriteriaFile);
                    {$I+}
                    if IOresult = 0 then
                       checked := true;
                 end;
              if (members.problem <> probname) or
                 (members.choice <> alternative)
                 then
                 write(TempProblemFile,
                       tempmember);
              end;    {While Statement}
     if checked then
        begin    {if checked}
           temp1 := probname+alternative+'.zzq';
           Assign(checkfile,concat(filedrive,':',
                 temp1));
           {$I-}
           erase(checkfile);
           {$I+}
```

```
                  if IOresult = 0 then
                     checked := true;
                  templ := probname+'.zzw';
                  Assign(checkfile,concat(filedrive,':',
                         templ));
                  {$I-}
                  erase(checkfile);
                  {$I+}
                  if IOresult = 0 then
                     checked := true;
                  templ := probname+alternative+'.zzx';
                  Assign(checkfile,concat(filedrive,':',
                         templ));
                  {$I-}
                  erase(checkfile);
                  {$I+}
                  if IOresult = 0 then
                     checked := true;
                  templ := probname+alternative+'.zzz';
                  Assign(checkfile,concat(filedrive,':',
                         templ));
                  {$I-}
                  erase(checkfile);
                  {$I+}
                  if IOresult = 0 then
                     checked := true;
               end;     {if checked}
            close(activeproblemfile);
            close(tempproblemfile);
            erase(activeproblemfile);
            rename(tempproblemfile,'probs.txt');
            if checked then
               begin
                  loadit;
                  gotoxy(12,14);
                  write('The Problem ',probname,'
                         has been deleted.');
                  delay(3000);
               end;
         end;     {if not stopprog}
         if not (checked) and
            not (stopprog) then
            warning;
      counts := succ(counts);
   until (checked) or (stopprog) or (counts > 2);
      end    {If the filesize statement}
   else
      nofiles;
end;    {DeleteAProblem}
```

```
procedure CheckAProblem;

(***********************************************************
 *  PROCEDURE           :  CHECKAPROBLEM                    *
 *  SUPPORTS PROGRAM    :  BTOUCH.PAS                       *
 *  LOCAL VARIABLES     :  SHORTNAME, COUNTS                *
 *  GLOBAL VARIABLES    :  STOPGAP, Z, INPUTSTRING, STOPPROG,*
 *                         PROBNAME, MEMBERS, ALTERNATIVE, Y *
 *  ARRAYS USED         :  NONE                             *
 *  FILES ACCESSED      :  ACTIVEPROBLEMFILE                *
 *  EXTERNAL CALLS      :  LOADIT, WARNING, NOFILES         *
 *  EXTERNAL FILTERS    :  FILTER9.LIB                      *
 *  CALLED FROM         :  PROBMANIPULATION                 *
 *  PURPOSE             :  GIVES AN INVOCATOR A DISPLAY OF  *
 *                         MEMBERS ON A SPECIFIC PROBLEM AND *
 *                         WHEN THAT MEMBER LAST ACCESSED   *
 *                         HIS PROBLEM.                     *
 ***********************************************************)

var
   SHORTNAME  :  STRING[7];
   COUNTS     :  INTEGER;

   begin   {CheckAProblem}
      counts := 0;      stopgap := false;      LoadIt;
      Reset(ActiveProblemFile);
      z := (filesize(activeproblemfile));
      close(activeproblemfile);
      if  z > 0 then
         begin   {If the filesize statement}
            repeat
               gotoxy(1,12);
               write('Entering a Problem name',
               ' from this list will tell you');
               gotoxy(1,13);
               write('When a member last',
               ' accessed a Problem');
               gotoxy(1,15);
               write('Enter the name of the Problem:   ');
               repeat
                  getthekeys(Inputstring,7);
                  shortName := inputstring;
                  gotoxy(33,16);
                  if stopprog then
                     stopgap := true;
               until (ord(shortname[1]) > 32) or (stopprog);
               a := 2;
               probname := shortName[1];
               while (shortname[a] <> chr(32)) and (a<8) do
                  begin
                     probname :=
                     concat(probname,shortname[a]);
                     a := a + 1;
                  end;
               y := 3;
            Reset(ActiveProblemFile);
            while not EOF(ActiveProblemFile) do
               begin   {While statement}
                  read(ActiveProblemFile,members);
                  if (members.problem = probname) and
                     (members.choice = alternative) then
                     begin
                        if Y = 3 then clrscr;
                        gotoxy(14,1);
                        write('PROBLEM    MEMBER
                               DATE');
                        stopgap := true;
                        gotoxy(14,Y);
                        write(members.problem);
                        gotoxy(25,Y);
                        write(members.member);
                        gotoxy(34,Y);
                        writeln(members.dateline);
                        Y := Y + 1;
                        if Y > 11 then
                           begin
                              gotoxy(16,16);
                              write('Press RETURN to
                                     continue');
                              getthekeys(inputstring,1);
                              clrscr;            Y := 3;
                           end;
                     end;
               end;     {While statement}
            close(activeproblemfile);
            if not (stopgap) and not (stopprog) then
               warning;
            counts := counts + 1;
         until (stopgap) or (counts > 2) or (stopprog);
         if not (stopprog) and (stopgap) then
                begin
                   gotoxy(16,16);
                   write('Press RETURN to continue.');
                   getthekeys(inputstring,1);
                end;
            end    {If the filesize statement}
   else
      nofiles;
end;    {CheckAProblem}
```

```
procedure DeleteAMember;

(*******************************************************
 *  PROCEDURE          :  DELETEAMEMBER                  *
 *  SUPPORTS PROGRAM   :  BTOUCH.PAS                     *
 *  LOCAL VARIABLES    :  SHORTNAME, FILECHECK, MAGGIE,  *
 *                        COUNTS, MARGARET, TEMPMEMBER   *
 *  GLOBAL VARIABLES   :  INPUTSTRING, Z, STOPGAP, STOPPROG,*
 *                        PROBNAME, ALTERNATIVE, MEMBERS, *
 *                        NEWSTRING                      *
 *  ARRAYS USED        :  NONE                           *
 *  FILES ACCESSED     :  TEMPPROBLEMFILE, ACTIVEPROBLEMFILE*
 *  EXTERNAL CALLS     :  FILTER9.LIB                    *
 *  EXTERNAL FILTERS   :  LOADIT, DISPLAYIT, NOFILES,    *
 *                        GETTHEKEYS                     *
 *  CALLED FROM        :  PERSMANIPULATION               *
 *  PURPOSE            :  THIS PROCEDURE ALLOWS THE USER TO *
 *                        SELECT A MEMBER AND PROBLEM FROM *
 *                        THE SCREEN FROM THE PROCEDURE  *
 *                        'LOADIT' FOR DELETEION.  THIS  *
 *                        WILL ONLY DELETE ONE MEMBER FOR *
 *                        THE SPECIFIC PROBLEM SELECTED. *
 ******************************************************)


var
   SHORTNAME           :  STRING[7];
   FILECHECK, MAGGIE   :  BOOLEAN;
   COUNTS              :  INTEGER;
   TEMPPROBLEMFILE     :  file of PROBREC;
   TEMPMEMBER          :  PROBREC;
   MARGARET            :  INTEGER;

   begin   {DeleteAMember}
      Reset(ActiveProblemFile);
      z := (filesize(activeproblemfile));
      close(activeproblemfile);
      if  z > 0 then
         begin   {If the filesize statement}
            repeat
               Margaret := 0;          Maggie := false;
               loadit;
               stopgap := false;       counts := 0;
               gotoxy(6,12);
               write('To quit without deleting a Member,
                     Press F10.');
               repeat
                  gotoxy(6,14);
                  write('Enter the Member''s PROBLEM:   ');
                  gotoxy(34,14);
                  repeat
                     getthekeys(Inputstring,7);
                     shortName := inputstring;
               gotoxy(34,14);
            until  (ord(shortname[1]) > 32) or
                   (stopprog);
            a := 2;       probname := shortName[1];
            while (shortname[a] <> chr(32)) and (a<3)
              do
               begin
                  probname :=
                  concat(probname,shortname[a]);
                  a := a + 1;
               end;
            reset(activeproblemfile);
            while not EOF(activeproblemfile) do
               begin   {While Statement}
                  read(ActiveProblemFile, members);
                  if (members.problem = probname) and
                     (members.choice = alternative)
                    then
                      margaret := succ(margaret);
               end;    {while statement}
            if margaret = 2 then
               begin
                  maggie := true;
                  counts := 3;
               end
            else
               stopgap := true;
            if not (stopgap) and not (stopprog) and
               not (maggie) then
               warning;
            counts := succ(counts);
         until (counts > 2) or (stopgap) or
               (stopprog);
         close(activeproblemfile);
         counts := 0;
         if (maggie) and not (stopprog) then
            begin
               gotoxy(1,15);  textbackground(red);
               write(' DELETION ABORTED! Committee ',
                   'would have less than 2 members ');
               delay(4000);
               textbackground(blue);  gotoxy(1,15);
               clreol;
               stopprog := true;
               stopgap := false;
            end;
         if stopgap then
            begin   {if stopgap}
               filecheck := false;
               displayit;
               repeat
                  gotoxy(1,14);
                  write('Enter the MEMBER',
                        ' initials that are to be
                          removed:  ');
```

```
repeat
   getthekeys(Inputstring,3);
   NewName := inputstring;
until (ord(shortname[1]) > 32) or
      (stopprog);
reset(ActiveProblemFile);
Assign(tempproblemfile,
concat(filedrive,':tempprob.txt'));
rewrite(tempProblemFile);
while not EOF(activeproblemfile) do
begin   {While Statement}
   read(ActiveProblemFile, members);
   tempmember := members;
   if (members.problem = probname)
      and
      (members.member = NewName)
      and
      (members.choice = alternative)
      then
       begin
          filecheck := true;
          NewString :=
          members.problem+
          alternative+
          '.'+members.member;
          Assign(kriteriafile,
          concat (filedrive,
                 ':',newstring));
          {$I-}
          erase(KriteriaFile);
          {$I+}
          if IOresult = 0 then
             stopgap := true;
       end;
   if (members.problem <> probname)
      or
      (members.member <> NewName)
      then
       write(TempProblemFile,
               tempmember);
end;    {While Statement}
close(activeproblemfile);
if not (filecheck) and
   not (stopprog) then
   begin
      gotoxy(14,15);
      textbackground(red);
      write(' Member is not on that
              committee ');
      delay(4000);
      textbackground(blue);
      gotoxy(15,15);
      clreol;
   end;
   if filecheck then
```

```
                                       begin
                                          clrscr;      gotoxy(1,9);
                                          write('The Member
                                                ',NewName,
                                          ' in the committee handling
                                            the problem');
                                          gotoxy(1,10);
                                          write(probname,' has been
                                            deleted.');
                                          delay(2000);
                                       end;
                                    counts := succ(counts);
                              until (counts > 2) or (filecheck) or
                                    (stopprog);
                              close(tempproblemfile);
                              erase(activeproblemfile);
                              rename(tempproblemfile,
                              concat(filedrive ,':probs.txt'));
                           end;   {if stopgap}
                     until stopprog;
                  end    {If the filesize statement}
               else
                  nofiles;
            end;    {DeleteAMember}

         procedure AddAMember;

         (************************************************************
          *  PROCEDURE          :  ADDAMEMBER                         *
          *  SUPPORTS PROGRAM   :  BTOUCH.PAS                         *
          *  LOCAL VARIABLES    :  TEMPNUM, SHORTNAME,                *
          *                        TEMPDEFINITION, CODE,              *
          *                        VERTZ, FILECHECK, TEMPNUMBER,      *
          *                        TEMPMEMBER                         *
          *  GLOBAL VARIABLES   :  Z, COUNT, LIMMIT, PROBNAME,        *
          *                        ALTERNATIVE, MEMBERS, STOPPROG,    *
          *                        INPUTSTRING, MOVEOVER, NEWSTRING   *
          *  ARRAYS USED        :  NONE                               *
          *  FILES ACCESSED     :  ACTIVEPROBLEMFILE, KRITERIAFILE    *
          *  EXTERNAL CALLS     :  LOADIT, WARNING, DISPLAYIT,        *
          *                        NOFILES, GETTHEKEYS                *
          *  EXTERNAL FILTERS   :  FILTER9.LIB                        *
          *  CALLED FROM        :  PERSMANIPULATION                   *
          *  PURPOSE            :  THIS PROCEDURE ALLOWS THE USER TO  *
          *                        SELECT A PROBLEM THAT IS ALREADY   *
          *                        ACTIVE AND ADD A MEMBER.  THE      *
          *                        USER IS ALLOWED TO VIEW ALL        *
          *                        PROBLEMS AND THE MEMBERS ON THAT   *
          *                        COMMITTE.                          *
          ************************************************************)
```

```
var
   TEMPNUM          :  STRING[2];
   SHORTNAME        :  STRING[7];
   TEMPDEFINITION :  STRING[59];
   CODE, VERTZ      :  INTEGER;
   FILECHECK        :  BOOLEAN;
   TEMPNUMBER       :  INTEGER;
   TEMPMEMBER       :  STRING3;

   begin   {AddAMember}
      LoadIt;
      filecheck := false;
      Reset(ActiveProblemFile);
      z := (filesize(activeproblemfile));
      close(activeproblemfile);
      if  z > 0 then
         begin   {If the filesize statement}
      gotoxy(1,12);
      Write('Please enter the name of the problem to which
             you');
      gotoxy(1,13);
      write('wish to add a member.  ');
      count := 0;      limmit := 0;
      repeat
         gotoxy(1,14);
         Write('The name must be listed above:   ');
         repeat
            getthekeys(Inputstring,7);
            shortName := inputstring;
            gotoxy(33,14);
         until (ord(shortname[1]) > 32) or (stopprog);
         a := 2;      probname := shortName[1];
         while (shortname[a] <> chr(32)) and (a<8) do
            begin
               probname := concat(probname,shortname[a]);
               a := a + 1;
            end;
         Reset(ActiveProblemfile);
         while not EOF(activeproblemfile) do
            begin   {while statement}
               Read(ActiveProblemFile,Members);
               if (Members.Problem = ProbName) and
                  (members.choice = alternative) then
                   begin
                      tempdefinition := members.definition;
                      limmit := limmit + 1;
                      filecheck := true;
                   end;
            end;    {while statement}
         close(ActiveProblemfile);
         if not (filecheck) and
            not (stopprog) then
                 warning;
         count := succ(count);
      until (filecheck) or (count > 2) or (stopprog);
```

```
if filecheck then
   begin  {if filecheck statement}
      displayit;
      repeat
         repeat
         gotoxy(1,15);
         Write('How many members do you',
         '  wish to add to this committee?  ');
         getthekeys(inputstring,2);
         tempnum := inputstring;
         gotoxy(56,15);
         val(tempnum,tempnumber,code);
         if (limmit + tempnumber > 15) then
            begin
               gotoxy(7,16);  textbackground(red);
               write(' There will be over 15',
               ' members on that committee ');
               delay(4000);
               textbackground(blue);
               gotoxy(7,16);  clreol;
               filecheck := false;
               stopprog := true;
            end;
         until (filecheck) or (stopprog);
      until (tempnumber > 0) and (tempnumber < 14) or
            (stopprog);
      if not stopprog then
         begin   {if not stopprog}
            moveover := 17;      count := 0;
            vertz := 15;
            GotoXY(1,15);      clreol;
            repeat
               limmit := 0;
               GotoXY(1,15);
               Write('Members names:  ');
               gotoxy(moveover,vertz);
               getthekeys(Inputstring,3);
               tempmember := inputstring;
               Reset(ActiveProblemfile);
               while not EOF(activeproblemfile) do
                  begin   {while statement}
                     Read(ActiveProblemFile,Members);
                     if (Members.member = tempmember)
                       and
                        (members.problem = probname)
                       and
                        (members.choice = alternative)
                       then
                         limmit := limmit + 1;
                     if tempmember = '    ' then
                        limmit := 100;
                  end;    {while statement}
               close(ActiveProblemfile);
               if (limmit = 0) and
                  not (stopprog) then
```

```
                                begin
                                   Members.Member := tempmember;
                                   Members.Checkstate := 'a';
                                   members.dateline := 'Empty File';
                                   members.definition :=
                                      tempdefinition;
                                   members.problem := probname;
                                   members.choice := alternative;
                                   reset(activeproblemfile);
                                   Seek(ActiveProblemFile,
                                   Filesize(ActiveProblemFile));
                                   Write(ActiveProblemfile,Members);
                                   close(ActiveProblemfile);
                                   NewString :=
                                   probname+alternative+
                                   '.'+members.member;
                                   Assign(kriteriafile,
                                   concat(filedrive,':',newstring));
                                   rewrite(Kriteriafile);
                                   close(Kriteriafile);
                                   moveover := moveover + 5;
                                   count := count + 1;
                                   if count = 8 then
                                      begin
                                         vertz := 16;
                                         moveover := 17;
                                      end;
                                end
                             else
                                if not stopprog then
                                   begin   {warning}
                                      gotoxy(12,13);
                                      textbackground(red);
                                      if limmit = 100 then
                                         write(' You must enter
                                           member''s initials ')
                                      else
                                         write(' Member is already
                                              on that committee ');
                                      delay(4000);
                                      textbackground(blue);
                                      gotoxy(12,13);  clreol;
                                   end;    {warning}
                          until (count = tempnumber) or (stopprog);
                          displayit;                    delay(4000);
                       end;    {if not stopprog}
                 end;   {if filecheck statement}
                 end    {If the filesize statement}
              else
                 nofiles;
           end;    {AddAMember}
```

```
procedure CheckforDoubles;

(***********************************************************
 *   PROCEDURE           :  CHECKFORDOUBLES                  *
 *   SUPPORTS PROGRAM    :  BTOUCH.PAS                       *
 *   LOCAL VARIABLES     :  NONE                             *
 *   GLOBAL VARIABLES    :  STARTUP, COUNT, MEMBERS,         *
 *                          ALTERNATIVE, PROBNAME            *
 *   ARRAYS USED         :  NONE                             *
 *   FILES ACCESSED      :  ACTIVEPROBLEMFILE = 'PROBS.TXT'  *
 *   EXTERNAL CALLS      :  NONE                             *
 *   EXTERNAL FILTERS    :  NONE                             *
 *   CALLED FROM         :  NEWPROBLEM                       *
 *   PURPOSE             :  THIS PROCEDURE PREVENTS THE      *
 *                          INVOCATOR FROM CREATING A PROBLEM *
 *                          WITH A DUPLICATE NAME, THEREBYE  *
 *                          OVERWRITING AN ACTIVE PROBLEM.   *
 *                          IT GIVES THE INVOCATOR THE       *
 *                          OPPORTUNITY TO RENAME THE 'NEW'  *
 *                          PROBLEM.  IF HE CHOOSES NOT TO   *
 *                          RENAME THE NEW PROBLEM, HE IS NOT *
 *                          ALLOWED TO CREATE IT             *
 ***********************************************************)


   begin    {CheckForDoubles}
      count := 1;       StartUp := false;
      Reset(ActiveProblemFile);
      while not EOF(ActiveProblemFile) do
         begin   {While statement}
            read(ActiveProblemFile,members);
            if (members.problem = probname) and
               (members.choice = alternative) then
                StartUp := true;
         end;   {while statement}
      close(activeproblemfile);
   end;     {CheckForDoubles}
```

```
procedure NewProblem;

(*************************************************************
*  PROCEDURE          :  NEWPROBLEM                          *
*  SUPPORTS PROGRAM   :  BTOUCH.PAS                          *
*  LOCAL VARIABLES    :  TEMPNUM, SHORTNAME, CODE,           *
*                        TEMPNUMBER, CHM, TEMPMEMBER,        *
*                        TEMPDEF                             *
*  GLOBAL VARIABLES   :  INPUTSTRING, ANONYMOUS, STOP, A,    *
*                        PROBNAME, CHATOK, STARTUP, CH,      *
*                        STOPPROG, MEMBERS, Y, MOVEOVER,     *
*                        COUNT, ALTERNATIVE, FILEDRIVE,      *
*  ARRAYS USED        :  NONE                                *
*  FILES ACCESSED     :  ACTIVEPROBLEMFILE,                  *
*  EXTERNAL CALLS     :  CHECKFORDOUBLES, GETTHEKEYS,        *
*                        INTROSCREEN, SETFILE                *
*  EXTERNAL FILTERS   :  FILTER1.LIB, FILTER7.LIB,           *
*                        FILTER9.LIB                         *
*  CALLED FROM        :  PROBMANIPULATION                    *
*  PURPOSE            :  ALLOWS THE INOVCATOR TO CREATE A    *
*                        NEW PROBLEM FOR EITHER              *
*                        ALTERANTIVES OR CRITERIA.           *
*************************************************************)


var
   TEMPNUM                    :  STRING[2];
   SHORTNAME                  :  STRING[7];
   CODE,TEMPNUMBER            :  INTEGER;
   CHM                        :  CHAR;
   TEMPMEMBER                 :  STRING3;
   TEMPDEF                    :  STRING[58];

   label 100;

   begin   {NewProblem}
      Anonymous := False;  {Stop := True;}      clrscr;
      Assign(ActiveProblemFile,
         concat(filedrive,':Probs.txt'));
      IntroScreen;              100:            GotoXY(2,2);
      Write('Please enter the name of the new problem.');
      GotoXY(2,3);
      Write('The name must not exceed seven letters:   ');
      gotoxy(50;3);
      repeat
         getthekeys(Inputstring,7);
         shortName := inputstring;
         gotoxy(50,3);
      until (ord(shortname[1]) > 32) or (stopprog);
      a := 2;                probname := shortName[1];
      while (shortname[a] <> chr(32)) and (a<8) do
         begin
            probname := concat(probname,shortname[a]);
            a := a + 1;
         end;
```

```
        CheckForDoubles;                ChatOK := True;

   (*************************************************************
    * AT THIS POINT THE PROGRAM HAS GONE AND CHECKED TO SEE IF*
    * THERE ARE ANY EXISTING PROBLEMS WITH THE SAME NAME .  IF*
    * THERE ARE, THEN THE BOOLEAN VARIABLE 'StartUp' IS SET TO*
    * TRUE AND THE NEXT 'IF' STATEMENT IS ACTIVATED.          *
    *************************************************************)
        if StartUp then
           begin   {Embedded If StartUp Statement Warning}
           window(6,4,74,21);      textbackground(red);
           clrscr;
           gotoxy(6,5);
           write('ATTENTION!!!  THERE IS A FILE ALREADY WITH
                  THE NAME ',probname);
           gotoxy(6,7);
           write('IN OUR FILES.  IN ORDER TO GO ON, YOU WILL
                  HAVE TO',' GIVE THIS');
           gotoxy(6,9);
           write('PROBLEM A NEW NAME OR DELETE THE OLD ONE.
                  DO YOU',' WISH TO');
           gotoxy(6,11);
           write('CONTINUE, GIVING THE NEW PROBLEM A DIFFERENT
                  NAME?','   Y/N');
           repeat
              gotoxy(66,11);
              getthekeys(Inputstring,1);
              Ch := inputstring;
              chm := ch;
           until ChM in ['Y','N'];
           if ch = #89 then
               begin
                  textbackground(blue);             clrscr;
                  window(12,5,73,20);    clrscr;  goto 100;
               end;
     end;    {Embedded If StartUp Statement Warning}

   (*************************************************************
    * AT THIS POINT THE PROGRAM HAS GONE AND CHECKED TO SEE   *
    * IF THERE ARE ANY EXISTING PROBLEMS WITH THE SAME NAME . *
    * IF THERE ARE NOT, THEN THE BOOLEAN VARIABLE 'StartUp'   *
    * IS SET TO FALSE AND THE NEXT IF STATEMENT IS ACTIVATED. *
    *************************************************************)

           if not (StartUp) and not (stopprog) then
              begin   {Embedded If not StartUp Statement}
                 Reset(ActiveProblemFile);
                 Seek(ActiveProblemFile,Filesize(ActiveProblemFile));
                 members.problem := probname;
                 GotoXY(2,4);
                 Writeln('Please give a one line definition of
                          the problem: ');
                 gotoxy(2,5);
                 getthekeys(Inputstring,58);
                 tempDef := inputstring;
```

```
gotoxy(2,6);
write('Do you wish to elaborate on that
       definition?  ');
repeat
   gotoxy(59,6);
   getthekeys(Inputstring,1);
   ch := inputstring;
   chm := ch;
until ChM in ['Y','N'];
if ch = 'Y' then scrollbox(13,11,51,'z');
window(12,5,73,20);  textbackground(blue);
gotoxy(2,7);
Write('How many members comprise this
       committee?  ');
repeat
   gotoxy(58,7);
   getthekeys(inputstring,2);
   tempnum := inputstring;
   val(tempnum,tempnumber,code);
    if tempnumber < 10 then
       begin
          gotoxy(58,7);
          clreol;
          write('  ');
          textbackground(yellow);
          write(tempnumber);
          textbackground(blue);
       end;
until (tempnumber > 1) and (tempnumber < 16);
GotoXY(2,8);   Write('Members names:   ');
count := 0;
Y := 8;
moveover := 57;
repeat
   limmit := 0;
   if Y > 10 then
      begin
         moveover := moveover - 8;
         Y := 8;
         GotoXY(2,8);
         Write('Members names:   ');
      end;
   repeat
      gotoxy(moveover,Y);
      getthekeys(Inputstring,3);
      shortName := inputstring;
      tempmember := inputstring;
   until ord(shortname[1]) > 32;
Reset(ActiveProblemfile);
while not EOF(activeproblemfile) do
   begin   {while statement}
      Read(ActiveProblemFile,Members);
      if (Members.member = tempmember) and
         (members.problem = probname) and
         (members.choice = alternative) then
```

```
            limmit := limmit + 1;
         end;     {while statement}
      close(ActiveProblemfile);
      if (limmit = 0) and
           not (stopprog) then
             begin
                Members.Member := tempmember;
                Members.Checkstate := 'a';
                members.dateline := 'Empty File';
                members.problem := probname;
                members.definition := tempdef;
                members.choice := alternative;
                members.checkchange := 'N';
                reset(activeproblemfile);
                Seek(ActiveProblemFile,
                Filesize(ActiveProblemFile));
                Write(ActiveProblemfile,Members);
                close(ActiveProblemfile);
                NewString := probname+alternative+
                                  '.'+members.member;
                Assign(kriteriafile,
                concat(filedrive,':',newstring));
                rewrite(Kriteriafile);
                close(Kriteriafile);
                count := count + 1;
                Y := Y + 1;
             end
         else
             begin   {warning}
                gotoxy(13,16);  textbackground(red);
                write('Member is already on that
                           committee');
                delay(4000);
                textbackground(blue);
                gotoxy(13,16);  clreol;
             end;    {warning}
      until (count = tempnumber) or (stopprog);
      if tempnumber = 2 then
         GotoXY(2,10)
      else
         gotoxy(2,11);
      write('Will communications and criteria be
              anonymous? ');
      repeat   {anonymous communications?}
         if tempnumber = 2 then
            gotoxy(59,10)
         else
            gotoxy(59,11);
         getthekeys(Inputstring,1);
         ch := inputstring;
         chm := ch;
      until ChM in ['Y','N'];
```

```
                If Ch = #89 then
                   begin
                      Anonymous := True;
                      setfile;
                   end;
             end;     {Embedded If not StartUp Statement}
   end;    {NewProblem}


procedure verifythename;

(************************************************************
 *   PROCEDURE            :  VERIFYTHENAME                      *
 *   SUPPORTS PROGRAM     :  BTOUCH.PAS                         *
 *   LOCAL VARIABLES      :  SHORTNAME, COUNTS                  *
 *   GLOBAL VARIABLES     :  STOPGAP, INPUTSTRING, PROBNAME,    *
 *                           MEMBERS, ALT, STOPPROG, FILECHECK,*
 *                           NEWNAME, PRINTONE                  *
 *   ARRAYS USED          :  NONE                               *
 *   FILES ACCESSED       :  ACTIVEPROBLEMFILE                  *
 *   EXTERNAL CALLS       :  DISPLAYIT, GETTHEKEYS              *
 *   EXTERNAL FILTERS     :  FILTER9.LIB                        *
 *   CALLED FROM          :  PRINTALTERNATIVES,PRINTCHATTERBOX *
 *   PURPOSE              :  CHECKS THE ACTIVEPROBLEMFILE AND   *
 *                           VERIFIES THAT A MEMBER IS ON A     *
 *                           CERTAIN COMMITTEE.                 *
 ************************************************************)


var
   SHORTNAME            :  STRING[7];
   COUNTS               :  INTEGER;

   begin    {verifythename}
      stopgap := false;      counts := 0;
      repeat     {till filename verified}
         gotoxy(35,16);
         repeat
            getthekeys(Inputstring,7);
            shortName := inputstring;
            gotoxy(35,16);
         until (ord(shortname[1]) > 32) or (stopprog);
         a := 2;      probname := shortName[1];
         while (shortname[a] <> chr(32)) and (a<8) do
            begin
               probname := concat(probname,shortname[a]);
               a := a + 1;
            end;
         reset(activeproblemfile);
         while not EOF(activeproblemfile) do
            begin   {While Statement}
               read(ActiveProblemFile, members);
               if (members.problem = probname) and
                  (members.choice = alt) then
                   stopgap := true;
            end;
         if not (stopgap) and not (stopprog) then warning;
         counts := succ(counts);
               {till filename verified}
      until (counts > 2) or (stopgap) or (stopprog);
      close(activeproblemfile);
      counts := 0;
      if (stopgap) and (printone) then
         begin   {if stopgap and printone}
            filecheck := false;                   displavit;
            repeat
               gotoxy(1,16);
               write('Enter the MEMBER initials
                      of the file:  ');
               gotoxy(43,16);
               repeat
                  getthekeys(Inputstring,3);
                  NewName := inputstring;
                  gotoxy(43,16);
               until (ord(newname[1]) > 32) or (stopprog);
               reset(ActiveProblemFile);
               while not EOF(activeproblemfile) do
                  begin   {While Statement}
                     read(ActiveProblemFile, members);
                     if (members.problem = probname) and
                        (members.member = NewName)  and
                        (members.choice = alt) then
                         begin
                            filecheck := true;
                            stopgap := true;
                         end;
                  end;    {While Statement}
               close(activeproblemfile);
               if not (filecheck) and
                  not (stopprog) then
                  begin
                     gotoxy(14,15);  textbackground(red);
                     write(' Member is not on
                              that committee ');
                     delay(4000);
                     textbackground(blue);
                     gotoxy(14,15);
                     clreol;
                  end;
               counts := succ(counts);
            until (counts > 2) or (filecheck) or (stopprog);
         end;   {if stopgap and printone}
end;     {verifythename}
```

```
procedure printalternatives;

(*************************************************************
 *  PROCEDURE          :  PRINTALTERNATIVES                  *
 *  SUPPORTS PROGRAM   :  BTOUCH.PAS                         *
 *  LOCAL VARIABLES    :  SHORTNAME, TEMPALT, ZCOUNT         *
 *  GLOBAL VARIABLES   :  PRINTONE, ALTERNATIVE, ALT,        *
 *                        STOPGAP, STOPPROG, NEWSTRING,      *
 *                        PROBNAME, FILEDRIVE, NEWNAME. Z,   *
 *                        CRITERIA, MEMBERS                  *
 *  ARRAYS USED        :  NONE                               *
 *  FILES ACCESSED     :  ACTIVEPROBLEMFILE, KRITERIAFILE    *
 *  EXTERNAL CALLS     :  LOADIT, VERIFYTHENAME, WARNING,    *
 *                        NOFILES                            *
 *  EXTERNAL FILTERS   :  PRINTER(EXTERNAL DEVICE)           *
 *  CALLED FROM        :  CHATMANIPULATION                   *
 *  PURPOSE            :  PRINTS FILES ON SPECIFIC MEMBERS   *
 *                        ON A COMMITTEE FOR ALTERNATIVES,   *
 *                        EITHER COMPLETED OR IN PROCESS.    *
 *************************************************************)


var
   SHORTNAME   :   STRING[7];
   TEMPALT     :   CHAR;
   ZCOUNT      :   INTEGER;

   begin   {printalternatives}
      printone := true;
      Reset(ActiveProblemFile);
      zcount := (filesize(activeproblemfile));
      close(activeproblemfile);
      repeat   {main repeat statement}
         tempalt := alternative;      alternative := alt;
         loadit;
         alternative := tempalt;
      if  zcount > 0 then
         begin   {If the filesize statement}
            repeat
               stopgap := false;
               gotoxy(1,12);
               write('Entering a Problem Name',
               ' from this list will print that');
               gotoxy(1,13);
               write('file for you');
               gotoxy(6,14);
               write('To quit without printing a file,
                      Press F10.');
               gotoxy(1,16);                     clreol;
               write('Enter the name of the Problem:');
               verifythename;
               {$I-}
            if (filecheck) and not (stopprog) then
               begin  {conditions are met}
                  newstring :=
                  concat(probname+alt+'.'+newname);
                  assign(kriteriafile,
                     filedrive+':'+newstring);
                  Reset(kriteriaFile);
                  z := filesize(kriteriafile);
                  if z > 0 then
                     begin   {if filesize}
                        writeln(lst,'PROBLEM IS
                                ',probname);
                        writeln(lst);
                        writeln(lst);
                        while not EOF(kriteriafile) do
                           begin   {While statement}
                              read(kriteriafile,
                                   criteria);
                              write(lst,
                               criteria.critname,':   ');
                              writeln(lst,
                               criteria.critdef);
                           end;    {While statement}
                     end;    {if filesize}
                     close(kriteriafile);
               end    {conditions are met}
            else
               begin
                  if not (stopprog) then
                  warning;
               end;
            {$I+}
            if IOresult = 0 then stopgap := true;
            if (z = 0) and not (stopprog) then
               begin  {if filesize else}
                  gotoxy(21,15);
                  write('file is empty');
                  delay(3000);
                  gotoxy(21,15);
                  clreol;
               end;   {if filesize else}
         until (stopprog) or (stopgap);
      end    {If the filesize statement}
   else
      nofiles;
   until stopprog;      {main repeat statement}
end;    {printalternatives}
```

```
procedure printchatterbox;

(************************************************************
 *  PROCEDURE          :  PRINTCHATTERBOX                   *
 *  SUPPORTS PROGRAM   :  BTOUCH.PAS                        *
 *  LOCAL VARIABLES    :  SHORTNAME, TEMPSTRING, TEMPALT,   *
 *                        COUNTS, ZCOUNT                    *
 *  GLOBAL VARIABLES   :  PRINTONE, ALTERNATIVE, ALT,       *
 *                        STOPGAP, FILEDRIVE, STOPPROG      *
 *  ARRAYS USED        :  NONE                              *
 *  FILES ACCESSED     :  TEXTFILE, ACTIVEPROBLEMFILE       *
 *  EXTERNAL CALLS     :  LOADIT, VERIFYTHENAME, WARNING,   *
 *                        NOFILES                           *
 *  EXTERNAL FILTERS   :  PRINTER(EXTERNAL DEVICE)          *
 *  CALLED FROM        :  CHATMANIPULATION                  *
 *  PURPOSE            :  PRINTS FILES ON SPECIFIC PROBLEMS *
 *                        WHERE THE MEMBERS HAVE UTILIZED   *
 *                        THE CHATTERBOX.                   *
 ************************************************************)

var
   SHORTNAME          :  STRING[7];
   TEMPSTRING         :  STRING[54];
   TEXTFILE           :  TEXT;
   TEMPALT            :  CHAR;
   COUNTS, ZCOUNT     :  INTEGER;

   begin   {printchatterbox}
      printone := false;
      Reset(ActiveProblemFile);
      zcount := (filesize(activeproblemfile));
      close(activeproblemfile);
      repeat   {main repeat statement}
      tempalt := alternative;       alternative := alt;
      loadit;                       alternative := tempalt;
      if  zcount > 0 then
         begin   {If the filesize statement}
            repeat
               stopgap := false;
               gotoxy(1,12);
               write('Entering a Problem Name',
               ' from this list will print that');
               gotoxy(1,13);
               write('file for you');
               gotoxy(6,14);
               write('To quit without printing a file,
                      Press F10.');
               gotoxy(1,16);                   clreol;
               write('Enter the name of the Problem:');
               verifythename;             counts := 0;
```

```
              if (stopgap) and not (stopprog) then
                 begin  {conditions are met}
                    NewString := probname+alt+'.zzz';
                    Assign(textfile,concat(filedrive,
                    ':',newstring));
                    {$I-}
                    Reset(textfile);
                    {$I+}
                    if IOresult = 0 then
                       begin  {IOresult}
                          writeln(lst,'CHATTERBOX IS
                                       ',probname);
                          writeln(lst);
                          writeln(lst);
                          while not EOF(textfile) do
                             begin   {While statement}
                                readln(textfile,
                                 tempstring);
                                writeln(lst,tempstring);
                                counts := succ(counts);
                             end;    {While statement}
                       end;   {IOresult}
                    close(textfile);
                 end    {conditions are met}
              else
                 begin
                    if not (stopprog) then
                    warning;
                 end;
              if (counts = 0) and not (stopprog) then
                 begin  {if filesize else}
                    gotoxy(21,15);
                    write('file is empty');
                    delay(3000);
                    gotoxy(21,15);
                    clreol;
                 end;  {if filesize else}
           until (stopprog) or (stopgap);
        end    {If the filesize statement}
     else
        nofiles;
     until stopprog;     {main repeat statement}
  end;    {printchatterbox}
```

```
{TAILEND.LIB}

procedure FinalChoice;

{*********************************************************
 *  PROCEDURE          :  FINALCHOICE                    *
 *  SUPPORTS PROGRAM   :  CTOUCH.PAS                     *
 *  LOCAL VARIABLES    :  NONE                           *
 *  GLOBAL VARIABLES   :  PROBLEMFLAG, FLAGCHOICE, COUNT, *
 *                        MEMBERS, NAMESTRING, PROBNAME, *
 *                        ALTERNATIVE                    *
 *  ARRAYS USED        :  NONE                           *
 *  FILES ACCESSED     :  ACTIVEPROBLEMFILE = 'PROBS.TXT' *
 *  EXTERNAL CALLS     :  NONE                           *
 *  EXTERNAL FILTERS   :  NONE                           *
 *  CALLED FROM        :  REVIEW, WINDOW3                *
 *  PURPOSE            :  IF THREE CONDITIONS ARE MET, THEN *
 *                        MEMBERS.  CHECKSTATE IS CHANGED *
 *                        TO WHATEVER THE NEW VALUE OF   *
 *                        PROBLEMFLAG IS LOADED INTO THAT *
 *                        RECORD.                        *
 *********************************************************}

   begin   {FinalChoice}

      case ProblemFlag of

         'a'  :  ProblemFlag := 'h';
         'b'  :  ProblemFlag := 'k';
         'c'  :  ProblemFlag := 'n';
         'd'  :  ProblemFlag := 'q';
         'i'  :  ProblemFlag := 'j';
         'l'  :  ProblemFlag := 'm';
         'o'  :  ProblemFlag := 'p';

      end;   {case statement}

         flagchoice := ' ';

         reset(ActiveProblemFile);

         Count := 1;

         while not EOF(ActiveProblemFile) do

            begin   {While Statement}

               read(ActiveProblemFile, members);

               if (members.member = namestring) and
                  (members.problem = probname) and
                  (members.choice = alternative) then
```

```
                    members.CheckState := problemflag;

                 seek(activeproblemfile,count-1);
                 write(activeproblemfile,members);
                 count := succ(count);

              end;    {While Statement}

           close(ActiveProblemFile);

     end;    {FinalChoice}


  procedure LoadArray;

  (**********************************************************
   *  PROCEDURE          :  LOADARRAY                        *
   *  SUPPORTS PROGRAM   :  CTOUCH.PAS                       *
   *  LOCAL VARIABLES    :  NONE                             *
   *  GLOBAL VARIABLES   :  TRACK1, LIMMIT, NAMES, Z         *
   *  ARRAYS USED        :  CRITARRAY                        *
   *  FILES ACCESSED     :  KRITERIAFILE                     *
   *  EXTERNAL CALLS     :  CRITSORT, NEWNUMBER, ODOMETER    *
   *  EXTERNAL FILTERS   :  NONE                             *
   *  CALLED FROM        :  WINDOW3                          *
   *  PURPOSE            :  LOADS THE ARRAY WITH THE USER'S  *
   *                        CHOSEN PROBLEM FOR RECORD        *
   *                        MANIPULATION BEFORE THE PROGRAM  *
   *                        TERMINATES.                      *
   **********************************************************)

     begin    {LoadArray}

        reset(Kriteriafile);
        z := filesize(kriteriafile);
        if z > 0 then
           begin   {if filesize}
              Track1 := 1;
              while not EOF(KriteriaFile) do
                 begin   {While Statement}
                    Read(KriteriaFile,Names[Track1]);
                    Track1 := Track1 + 1;
                 end;    {While Statement}

              Limmit := Track1;

           end;    {if filesize}
        close(KriteriaFile);

        CritSort(Names,Limmit);    NewNumber(Names,Limmit);
        Odometer;


     end;    {LoadArray}
```

```
procedure NewWrite(var Names  :  CritArray;  Limmit :
Integer);


(*************************************************************
 *  PROCEDURE           :  NEWWRITE                          *
 *  SUPPORTS PROGRAM    :  CTOUCH.PAS                        *
 *  LOCAL VARIABLES     :  NONE                              *
 *  GLOBAL VARIABLES    :  Z, TRACK1, NAMES, PROBLEMFLAG,    *
 *                         LIMMIT                            *
 *  ARRAYS USED         :  CRITARRAY                         *
 *  FILES ACCESSED      :  NONE                              *
 *  EXTERNAL CALLS      :  CRITSORT, NEWNUMBER               *
 *  EXTERNAL FILTERS    :  FILTER6.LIB                       *
 *  CALLED FROM         :  WINDOW3, REVIEW                   *
 *  PURPOSE             :  RELOADS THE CRITERIA FILE FROM    *
 *                         THE ARRAY THAT HAS BEEN CHANGED   *
 *                         THROUGH THE ACTIONS OF THE USER.  *
 ************************************************************)

   begin   {NewWrite}

      if z > 0 then

         begin   {if filesize}

            CritSort(Names,Limmit);      NewNumber(Names,
            Limmit);

            rewrite(Kriteriafile);

            Track1 := 1;

            repeat

               case names[Track1].flag1 of

                  1..100  :  begin
                                Names[Track1].StatFlag :=
                                   problemFlag;
                                Write(kriteriafile,
                                       Names[Track1]);
                             end;

               end;    {case statement}

               Track1 := Track1 + 1;

            until  (Track1 = Limmit);

         end;     {if filesize}

         close(KriteriaFile);
   end;     {NewWrite}


procedure ChangeRecord(var Names  :  CritArray;
                           Limmit : Integer);


{**********************************************************
 *  PROCEDURE            :  CHANGERECORD                  *
 *  SUPPORTS PROGRAM     :  CTOUCH.PAS                    *
 *  LOCAL VARIABLES      :  WRONGLEVEL, WRONGWORD, CHANGECRIT,*
 *                          CHM, TEMPALT, LONGNAME, SHORTNAME *
 *  GLOBAL VARIABLES     :  TRACK1, CHOICE, WITHOUTACHANGE,   *
 *                          FINDCODE, CH, PROBLEMFLAG, NAMES, *
 *                          STOPPROG, COUNTED             *
 *  ARRAYS USED          :  CRITARRAY                     *
 *  FILES ACCESSED       :  NONE                          *
 *  EXTERNAL CALLS       :  GETTHEKEYS, NEWNUMBER         *
 *  EXTERNAL FILTERS     :  FILTER6.LIB, FILTER9.LIB      *
 *  CALLED FROM          :  REVIEW                        *
 *  PURPOSE              :  ALLOWS THE USER TO CHANGE THE *
 *                          ALTERNATIVES/CRITERIA, AT THE *
 *                          LEVEL OF DEVELOPMENT THEY ARE *
 *                          AT.  WILL NOT ALLOW THEM TO   *
 *                          CHANGE CRITERIA AT A LEVEL    *
 *                          PREVIOUSLY FLAGGED AS FINISHED.  *
 **********************************************************}

var
   WrongLevel, WithoutAChange, FindCode  :  Boolean;
   WrongWord                             :  Boolean;
   changecrit                            :  string10;
   chm                                   :  char;
   shortname                             :  string10;
   longname                              :  string[38];
   tempalt                               :  string[12];

   begin   {ChangeRecord}

      if alternative = 'A' then
         tempalt := 'Alternative'
      else
         tempalt := 'Criteria';

      track1 := 0;                  choice :=  ' ';
      WithoutAChange := True;   Findcode := False;
      WrongLevel := True;
      gotoxy(2,2);              clreol;
      write('Enter the ',tempalt,' Name you wish to change
             or delete:  ');
      gotoxy(63,2);

      repeat
         getthekeys(Inputstring,10);
```

```
    shortName := inputstring;
    gotoxy(63,2);
until (ord(shortname[1]) > 32) or (stopprog);

a := 2;
changecrit := shortName[1];

while (shortname[a] <> chr(13)) and (a<11) do
    begin
        changecrit := concat(changecrit,shortname[a]);
        a := a + 1;
    end;

repeat
    gotoxy(2,2);          clreol;
    wronglevel := true;
    wrongword := false;
    track1 := track1 + 1;

    case problemflag of

        'a', 'i'   :  begin
                         if (names[track1].critname =
                              changecrit) and
                             (names[track1].flag2 = 0)
                          then
                            begin
                                WithoutAChange := False;
                                gotoxy(2,2);
                                write(Names[Track1].
                                        CritName,':   ',
                                Names[Track1].CritDef);
                                gotoxy(2,4);
                                write('Do you wish to
                                          delete this   ',
                                 'or change it?  D/C   ');
                                FindCode := True;
                                getthekeys(Inputstring,1);
                                choice := inputstring;
                                WrongLevel := False;
                                gotoxy(2,4);
                                clreol;
                            end;
                         if (names[track1].critname <>
                              changecrit) and
                             (names[track1].flag2 = 0)
                           then
                              wrongword := true;

                      end;

        'o', 'l'   :  begin
                         if (names[track1].critname =
                              changecrit) and
                             (names[track1].flag2   0) and
```

```
                    (names[track1].flag3 = 0)
                     then
                     begin
                         WithoutAChange := False;
                         gotoxy(2,2);
                         write(Names[Track1].
                               CritName,':  ',
                         Names[Track1].CritDef);
                         gotoxy(2,4);
                         write('Do you wish to
                                delete this ',
                         'or change it?  D/C  '),
                         FindCode := True;
                         getthekeys(Inputstring,1);
                         choice := inputstring;
                         WrongLevel := False;
                         gotoxy(2,4);
                         clreol;
                     end;
                 if (names[track1].critname <>
                     changecrit) and
                    (names[track1].flag3 = 0)
                   then
                     wrongword := true;
             end;

'c', 'o'  :  begin
                 if (names[track1].critname =
                     changecrit) and
                    (names[track1].flag3 > 0)
                   then
                     begin
                         WithoutAChange := False;
                         gotoxy(2,2);
                         write(Names[Track1].
                               CritName,':  ',
                         Names[Track1].CritDef);
                         gotoxy(2,4);
                         write('Do you wish to
                         delete this ',
                         'or change it?  D/C  ');
                         FindCode := True;
                         getthekeys(Inputstring,1);
                         choice := inputstring;
                         WrongLevel := False;
                         gotoxy(2,4);
                         clreol;
                     end;
                 if (names[track1].critname <>
                     changecrit) and
                    (names[track1].flag3   0)
                   then
                     wrongword := true;
             end;
```

```
          end;    {case statement}

   until (track1 = limmit-1) or (findcode);

   if wrongword then

      begin

         if alternative = 'A' then
            tempalt := 'Alternative'
         else
            tempalt := 'Criteria';

         clrscr;    sound(500);  delay(100);  nosound;
         gotoxy(13,2);
         writeln('You may have misspelled the ',tempalt,
              '. Try again.');
         delay(5000);
         gotoxy(13,2);              clreol;
         FindCode := True;
         wronglevel := false;

      end;

   if wronglevel then

      begin

         if alternative = 'A' then
            tempalt := 'Alternative'
         else
            tempalt := 'Criteria';

         clrscr;    sound(500);  delay(100);  nosound;
         gotoxy(12,2);
         writeln('You may not change the ',tempalt,
              ' at that level');
         delay(5000);
         gotoxy(12,2);              clreol;
         FindCode := True;
         wronglevel := false;

      end;

   if choice = 'D' then

      begin    {If Delete Statement}

         clrscr;
         gotoxy(2,2);
         write(Names[Track1].CritName,':   ',
         Names[Track1].CritDef);
         gotoxy(2,4);
         textbackground(red);
```

```
choice := ' ';      gotoxy(12,9);

if alternative = 'A' then
   tempalt := 'ALTERNATIVE'
else
   tempalt := 'CRITERIA';

write(' YOU ARE ABOUT TO DELETE THIS RECORD!!!
        BE ADVISED   ');
gotoxy(12,10);
write(' THAT A YES ANSWER TO THIS QUESTION WILL
       REMOVE THIS ');
gotoxy(12,11);
write(' ',TEMPALT,' PERMANENTLY.  DO YOU STILL
       WISH TO DELETE  ');
gotoxy(12,12);
write(' THIS ',tempalt,'?  Y/N
       ');
gotoxy(62,12);

repeat
   getthekeys(Inputstring,1);
   ch := inputstring;
   chm := ch;
   gotoxy(64,12);
until chm in ['Y','N'];

   clrscr;

if ch = 'Y' then

   begin    {Embedded If Delete Statement}

      ch := 'N';
      gotoxy(2,2);
      write(Names[Track1].CritName,':    ,
      Names[Track1].CritDef);
      gotoxy(2,4);
      gotoxy(21,11);
      write('This ',tempalt,' has been
             deleted');
      Names[Track1].Flag1 := 0;
      delay(4000);
      gotoxy(2,2);                        clreol;
      gotoxy(2,4);                        clreol;
      changerec := 'C';

   end;      {Embedded If Delete Statement}

end;      {If Delete Statement}

   if choice = 'C' then

      begin    {If Change Statement}
```

```
if alternative = 'A' then
   tempalt := 'Alternative'
else
   tempalt := 'Criteria';

    choice := ' ';
    gotoxy(2,3);
    write('Enter the New ',tempalt,' Name:   ');
    gotoxy(33,3);

    repeat
       getthekeys(Inputstring,10);
       shortName := inputstring;
       gotoxy(33,3);
    until (ord(shortname[1]) > 32) or
          (stopprog);
    a := 2;
    names[track1].critname := shortName[1];

    while (shortname[a] <> chr(13)) and
          (a<11) do
       begin
          names[track1].critname :=
          concat(names[track1].critname,
            shortname[a]);
          a := a + 1;
       end;

    gotoxy(2,4);   write('Definition:   ');

    gotoxy(15,4);

    repeat
       getthekeys(Inputstring,58);
       longName := inputstring;
       gotoxy(15,4);
    until (ord(longname[1]) > 32) or
          (stopprog);
    a := 2;
    names[track1].critdef := longName[1];

    while (longname[a] <> chr(13)) and
          (a<counted+1) do
       begin
          names[track1].critdef :=
          concat(names[track1].critdef,longname[a]);
          a := a + 1;
       end;

    clrscr;                      gotoxy(2,2);
    write(names[Track1].critname,' : ',
    names[Track1].critdef);

    gotoxy(22,4);
    write('The ',tempalt,' has been
                    changed');
               delay(2500);
               gotoxy(22,4);                    clreol;
               gotoxy(2,2);                     clreol;
               changerec := 'C';

            end;     {If Change Statement}

      NewNumber(Names, Limmit);

   end;    {ChangeRecord}

procedure RanToCompletion;

(**************************************************************
 *  PROCEDURE           :  RANTOCOMPLETION                    *
 *  SUPPORTS PROGRAM    :  CTOUCH.PAS                         *
 *  LOCAL VARIABLES     :  NONE                               *
 *  GLOBAL VARIABLES    :  I                                  *
 *  ARRAYS USED         :  NONE                               *
 *  FILES ACCESSED      :  NONE                               *
 *  EXTERNAL CALLS      :  PORT[$03D9], SETBORDER (INTERAL    *
 *                         PROCEDURE)                         *
 *  EXTERNAL FILTERS    :  NONE                               *
 *  CALLED FROM         :  WINDOW3                            *
 *  PURPOSE             :  THIS PROCEDURE INFORMS THE USER    *
 *                         THAT ALL MEMBERS OF THE COMMITTEE *
 *                         ARE IN COMPLETE AGREEMENT WITH     *
 *                         THE CRITERIA CONCERNING THE        *
 *                         PROBLEM.  IT DIRECTS THEM TO GO    *
 *                         ON TO THE FIRST STAGE OF THE       *
 *                         CO-OP SYSTEM. A BIT MUCH ISN'T IT?*
 *************************************************************)


procedure Setborder(color:byte);

   begin  {setborder}

      port[$03d9]:= $f and color;

   end:  {setborder}

   begin  {RanToCompletion}

      introscreen;
      gotoXY(8,8);
      write('You are now ready to enter the CO-OP system');
      gotoxy(18,16);
      write('Press any key to exit');
      repeat
         for I := 0 to 15 do
            begin
```

```
                setborder(I);
                delay(500);
             end;
        until keypressed;
        setborder(8);

     end;   {RanToCompletion}


procedure Review(var Names  :  CritArray;
                     Limmit : Integer);

(**************************************************************
 *  PROCEDURE           :  REVIEW                             *
 *  SUPPORTS PROGRAM    :  CTOUCH.PAS                         *
 *  LOCAL VARIABLES     :  CHM, TEMPALT                       *
 *  GLOBAL VARIABLES    :  PT1, PT2, PT3, PT4, PROBLEMFLAG,   *
 *                         SCROLLIT TRACK1, CH, INPUTSTRING,  *
 *                         FLAGCHOICE                         *
 *  ARRAYS USED         :  NONE                               *
 *  FILES ACCESSED      :  NONE                               *
 *  EXTERNAL CALLS      :  REVIEW1, GETTHEKEYS, CHANGERECORD,*
 *                         FINALCHOICE, NEWWRITE              *
 *  EXTERNAL FILTERS    :                                     *
 *  CALLED FROM         :                                     *
 *  PURPOSE             :  ALLOWS THE USER TO REVIEW PAST     *
 *                         ALTERNATIVES/CRITERIA, AND CHANGE *
 *                         THEM, DEPENDING AT WHAT STAGE OF   *
 *                         THE DEVELOPMENT THEY ARE AT.       *
 **************************************************************)

var
   CHM : CHAR;
   TEMPALT  : STRING[12];

   begin   {Review}

      clrscr;
      pt1 := 2;  pt2 := 2;  pt3 := 77;  pt4 := 21;
      window(pt1,pt2,pt3,pt4);  clrscr;

      scrollit := true;   track1 := 1;
      review1(names,limmit);

      case problemflag of

         'a'..'d','i','l','o'  :

            begin   {Inside of Case Statement}

               repeat

                  if alternative = 'A' then
                     tempalt := 'Alternatives'
                  else
```

```
                   tempalt := 'Criteria';

                gotoxy(12,1);   ch := 'N';  clreol;
                write('Do you Wish to Change a portion of
                       the ',tempalt,'?');
                gotoxy(12,3);   clreol;
                write('Press Home Key to activate
                       Scrolling.  Press Enter');
                gotoxy(12,4);   clreol;
                write('Key before answering the question
                       after Scrolling.');
                gotoxy(66,1);

                getthekeys(Inputstring,1);
                ch := inputstring;

                if ch = 'Y' then   { Y }

                   begin   {Embedded If Statement}
                      gotoxy(12,1);   clreol;
                      gotoxy(12,3);   clreol;
                      gotoxy(12,4);   clreol;
                      ch := 'N';
                      ChangeRecord(Names, Limmit);

                      track1 := 1;
                      review1(names,limmit);
                      Track1 := 1;

                   end;     {Embedded If Statement}

             until ch = 'N';

             scrollit := false;

          end;    {Inside Case Statement}

       end;  {Case Statement}

    case problemflag of

       'a'..'d','i','l','o'  :

          begin   {Inside of Case Statement}

             clrscr;   gotoxy(20,8);
             write('Are you finished reviewing this
                    level');
             gotoxy(20,9);
             write('or will there be more changes?
                    Enter ');
             gotoxy(20,10);
             write('''F'' for Finished or
                   ''M'' for More:  ');
             gotoxy(58,10);
```

```
            repeat
               getthekeys(Inputstring,1);
               flagchoice := inputstring;
               chm := flagchoice;
               gotoxy(53,10);
            until CHM in ['F','M'];

            if (FlagChoice = 'F') then
                FinalChoice;

         end;    {Inside of Case Statement}

      'h','j','k','m','n','p'  :
                       begin
                          gotoxy(2,2);
                          write('Press Return to
                                 continue:  ');
                          getthekeys(Inputstring,1);
                       end;

      end;    {case statement}

   NewWrite(Names,Limmit);

end;    {Review}
```

LIST OF REFERENCES

1. Quade, G. S., and Boucher, W. I., An Extended Concept of Model (P4427), Santa Monica, California, pp. 4-5, Rand, 1970.

2. Sprague, R. H., and Carlson, E. D., Building Effective Decision Support Systems, pp. 274-276, Prentice-Hall, Inc., New Jersey, 1982.

3. Bui, X. T., and Jarke, M., Communications Requirements for Group Decision Support Systems, working paper, Naval Postgraduate School, Monterey, California, 1986.

4. Heider, F., The Psychology of Interpersonal Relations, pp. 244-251, John Wiley & Sons, Inc., New York, 1958.

BIBLIOGRAPHY

Carlson, E. D., "Decision Support Systems: Personal Computing Services for Managers," Management Review, pp. 4-11, January 1977.

Christos, S., CO-OP 2.0, Distributed Decision Support System for Strategic Planning, Master's Thesis, Naval Postgraduate School, Monterey, California, March 1986.

Gallupe, R. B., "Experimental Research Into Group Decision Support Systems: Practical Issues and Problems," Proceedings of the Nineteenth Annual Hawaii International Conference on System Sciences, 1986, pp. 515-523, 1986.

Huber, G. P., "Issues in the Design of Group Decision Support Systems," MIS Quarterly, V. 8, No. 3, September, 1984.

Jarke, M., Bui, X. T., and Jelassi, M. T., "Micro-Mainframe DSS for Remote Multi-Person Decisions," Managers, Micros, and Mainframes, John Wiley & Sons Ltd., 1986.

Linstone, H. A., and Turoff, M., editors, The Delphi Method: Techniques and Applications, Addison-Wesley Publishing Company, 1975.

Rossy, G. L., Management By Objectives: A Forecast of Its Future Development Using the Delphi Technique, Ph.D. Dissertation, University of California, Los Angeles, California, 1979.

Suchan, J., Bui, T., and Dolk, D., GDSS Effectiveness: Identifying Organizational Opportunities, working paper, Naval Postgraduate School, Monterey, California, July 1986.

INITIAL DISTRIBUTION LIST

No. Copies

1. Commanding Officer 4
   Naval Health Sciences Education
   and Training Command (Code 34)
   Naval Medical Command
   National Capitol Region
   Bethesda, MD 20814

2. Naval Medical Data Services Center 1
   Naval Medical Command
   National Capitol Region
   Bethesda, MD 20814

3. Library, Code 0142 2
   Naval Postgraduate School
   Monterey, CA 93943-5002

4. Dr. X. Tung Bui, Code 54BD 2
   Department of Administrative Sciences
   Naval Postgraduate School
   Monterey, CA 93943-5000

5. Dr. Nancy Roberts, Code 54RC 1
   Department of Administrative Sciences
   Naval Postgraduate School
   Monterey, CA 93943-5000

6. CDR Robert T. Wooldridge, NC, USN 4
   Quality Assurance Unit
   Naval Hospital
   San Diego, CA 92134

7. LT Michael E. Neeley, MSC, USN 2
   Management Information Department
   Naval Hospital
   Pensacola, FL 32512-5000

8. Computer Technology Programs, Code 37 1
   Naval Postgraduate School
   Monterey, CA 93943-5000

9. Defense Technical Information 2
   Cameron Station
   Alexandria, VA 22304-6145

END
9-87
DTIC